PEMBANGUNAN SISTEM UNTUK PENDETEKSIAN CODE SMELLS REFUSED BEQUEST

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Faishal Firdaus
NIM: 145150201111094

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PEMBANGUNAN SISTEM UNTUK PENDETEKSIAN CODE SMELLS REFUSED BEQUEST

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Falshal Firdaus
NIM: 145150201111094

Skripsi ini telah diuji dan dinyatakan lulus pada
31 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Bayu Priyambadha, S.Kom, M.Kom	Fajar Pradana, S.ST, M.Eng
NIP: 19820909 200812 1 004	NIP: 19871121 201504 1 004

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 31 Juli 2018

Muhammad Faishal Firdaus
NIM: 145150201111094

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena atas segala limpahan kasih dan sayang-Nya penulis dapat menyelesaikan skripsi yang berjudul "PEMBANGUNAN SISTEM UNTUK PENDETEKSIAN *CODE SMELLS REFUSED BEQUEST*". Melalui pengantar ini, penulis hendak mengucapkan terimakasih karena dalam penyusunan skripsi ini, penulis telah mendapat bantuan dan dorongan dari berbagai pihak. Untuk itu pada kesempatan ini penulis hendak mengucapkan terima kasih kepada:

1. Kedua orang tua tercinta beserta saudara penulis atas segala doa, semangat, kasih sayang, dan dukungan yang selalu diberikan dengan tulus kepada penulis.
2. Bapak Bayu Priyambadha, S.Kom, M.Kom selaku Dosen Pembimbing I skripsi, yang telah mengarahkan, membina, dan membimbing saya sehingga dapat menyelesaikan skripsi ini.
3. Bapak Fajar Pradana, S.ST, M.Eng selaku Dosen Pembimbing II skripsi, yang telah mengarahkan, membina, dan membimbing saya sehingga dapat menyelesaikan skripsi ini.
4. Para Dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah bersedia membagi ilmu kepada penulis beserta Staff Akademik yang telah memberikan bantuan akademik selama proses menempuh studi hingga penyelesaian skripsi.
5. Teman-teman dari TPOC yang selalu memberi bantuan, motivasi, dan kebersamaan selama ini.
6. Teman-teman teknik informatika 2014 atas bantuan, motivasi dan kebersamaan selama ini sehingga penulis dapat menyelesaikan skripsi dengan baik.
7. Semua pihak yang tidak semuanya bisa dituliskan disini yang terlibat secara langsung maupun tidak langsung dalam proses pengerjaan skripsi maupun sebagai pemberi semangat dan motivasi.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 31 Juli 2018

Penulis
faishal@student.ub.ac.id

ABSTRAK

Code smells merupakan suatu karakteristik dari sebuah perangkat lunak yang mengindikasikan permasalahan pada struktur kode dan desain sistem yang mengakibatkan perangkat lunak tersebut sulit untuk dikembangkan dan dilakukan perawatan. Salah satu jenis *code smells* yang cukup terkenal adalah *refused bequest*, yang merupakan kondisi pada konsep pewarisan yaitu *subclass* tidak menggunakan fungsionalitas turunan dari *superclass* sehingga seolah-olah terjadi penolakan pewarisan. Umumnya, *code smells* dapat diidentifikasi melalui struktur kode program. Namun, pada penelitian ini dikembangkan pendeteksian *code smells* pada tahap pengembangan perangkat lunak yaitu perancangan. Perancangan merupakan fase yang sangat penting dalam tahapan pengembangan perangkat lunak karena keberhasilan sebuah perangkat lunak bergantung kepada analisa dan perancangan yang baik. Pada tahap perancangan yang dilakukan pendeteksian *code smells* dengan jenis *refused bequest* adalah pada perancangan komponen, yaitu perancangan *class diagram*. Rancangan *class diagram* dalam format .vpp dari aplikasi *UML Creator Visual Paradigm* diubah kedalam bahasa *xml*. Setelah diubah, file *xml* dideteksi pada perangkat lunak pendeteksian dengan melakukan *parsing* dan menemukan tingkatan *code smells refused bequest* yang mengacu kepada *thermometer smells* sebagai pengukur intensitas *refused bequest* yang ditemukan. Pembangunan sistem ini mengikuti tahapan pengembangan perangkat lunak yang dimulai dari tahap analisis kebutuhan, perancangan dan implementasi, serta pengujian sistem. Pengujian yang dilakukan pada sistem ini menggunakan pengujian *whitebox testing* untuk pengujian unit dan integrase serta *blackbox testing* untuk pengujian validasi.

Kata kunci: *code smells, refused bequest*, perancangan *class diagram*, sistem pendeteksian

ABSTRACT

Code smells are a characteristic of a software that indicates problems in the code structure and system design that result in the software is being difficult to develop and maintain. The kind of quite famous code smells is refused bequest, which is a condition in the concept of inheritance that subclasses do not use the derived functionality of the superclass so as to happen inheritance rejection. Generally, code smells only can be identified through the program code structure. However, in this study developed the detection code smells in the stage of software development is design. Design is a very important phase in the software development phase because the success of a software depends on good analysis and design. At the design stage performed detection code smells with the kind of refused bequest is on the design of components, namely class diagrams design. The design of the class diagram in the .vpp format of the UML Creator Visual Paradigm application is converted into the xml language. Once converted, the xml file is detected on the detection software by parsing and finding the code smells refused bequest level referring to the smells thermometer as the measured refused bequest intensity meter. Development of this system follows the stages of software development starting from the needs analysis phase, design and implementation, as well as system testing. The tests performed on this system use testing whitebox testing for unit testing and integrase and blackbox testing for validation testing.

Kata kunci: *code smells, refused bequest*, *class diagram design*, detection system.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Dalam konsep rekayasa perangkat lunak terdapat fase-fase pengembangan atau siklus hidup pengembangan sistem. Siklus tersebut dimulai dari fase analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian sistem, serta perawatan sistem. Proses pengembangan perangkat lunak merupakan serangkaian aktifitas untuk membangun suatu perangkat lunak dimana proses pembangunan ini dimulai dari dasar dengan menggunakan bahasa pemrograman yang sesuai (Sommerville, 2011). Salah satu fase pengembangan yang sangat penting adalah fase perancangan sistem, dimana keberhasilan suatu sistem bergantung kepada analisa dan perancangan sistem yang baik (Waldo, 2006). Tahap perancangan sistem merupakan acuan untuk melakukan proses implementasi pembangunan sistem atau proses *coding* sistem. Adapun pada fase pengembangan perangkat lunak terdapat fase perawatan sistem yang digunakan untuk menjamin kualitas dari perangkat lunak yang telah dibangun, serta menemukan beberapa *code smells* yang terdapat dalam struktur sistem.

Code smells merupakan suatu karakteristik dari perangkat lunak yang mengindikasikan permasalahan pada kode dan desain sistem sehingga mengakibatkan perangkat lunak sulit untuk dikembangkan dan dilakukan perawatan (Fontana, 2011). Pada skripsi ini akan dilakukan penelitian tentang salah satu jenis *code smells* yaitu *refused bequest*. *Refused bequest* merupakan suatu kondisi dimana pada konsep pemrograman berorientasi obyek yaitu pewarisan tidak digunakan sebagaimana mestinya, dimana *subclass* tidak menggunakan fungsionalitas turunan dari *superclass* sehingga *subclass* hanya menggunakan beberapa method yang diwariskan (Ligu, 2013). Hal ini mengakibatkan kemunculan *smells* dalam struktur program yang telah diidentifikasi.

Pada saat ini, banyak penelitian yang dilakukan terkait *code smells* hanya dalam struktur kode program saja. Seperti penelitian yang dilakukan oleh Hanson dalam jurnalnya "*Deteksi Code Smell Pada Kode Program Dalam Representasi Ast Dengan Pendekatan by Rules*". Pada penelitian Hanson yang dilakukan, dihasilkan sebuah perangkat deteksi *code smells* yang menggunakan masukan berupa *Abstract Syntax Tree* (AST) dari kode program dan juga kriteria yang telah disusun dengan pendekatan *by rules*. Menurut hasil penelitian Hanson, diketahui bahwa deteksi *code smells* pada level AST dapat dilakukan dalam semua bahasa pemrograman. Sehingga pada penelitian ini akan diusulkan suatu sistem yang dapat melakukan deteksi *code smells* dengan jenis *refused bequest* pada tahap perancangan sistem untuk mendapatkan hasil deteksi dari tahap awal pengembangan sistem. Proses awal untuk mendeteksi *refused bequest* adalah dengan melakukan *parsing class* diagram dari aplikasi *UML Creator Visual Paradigm* dengan format .vpp kedalam bahasa XML. Setelah dilakukan *parsing*, file.XML dimasukkan kedalam sistem deteksi lalu selanjutnya dianalisis oleh sistem, sehingga sistem dapat menemukan *refused bequest* yang terdapat dalam *class diagram* tersebut. Proses analisis yang dilakukan oleh sistem deteksi

didapatkan dari jurnal Elvis Ligu dengan judul "*Identification of Refused Bequest Code smells*" yang merupakan jurnal utama dari penelitian ini sebagai acuan dalam melakukan penggalian kebutuhan dan penelitian itu sendiri.

Pada penelitian ini telah diidentifikasi beberapa permasalahan, pada pendeteksian *code smells* belum terdapat sistem yang dapat mendeteksi *refused bequest* pada tahap perancangan sistem secara otomatis. Beberapa sistem telah dikembangkan, namun tidak mencakup keseluruhan jenis-jenis *code smells* yang ada. Pada permasalahan yang telah diuraikan, ditawarkan sebuah solusi untuk melakukan pendeteksian *code smells refused bequest* berupa "*Pembangunan Sistem untuk Pendeteksian Code smells Refused Bequest*". Dengan adanya sistem ini, pada saat melakukan perancangan *class* diagram diharapkan hasil perancangan tersebut telah bebas dari *code smells*, sehingga pada tahap implementasi proses pembangunan suatu sistem menjadi lebih bersih dari *code smells* yang dapat mengganggu proses perawatan sistem yang dibangun nantinya. Dengan adanya penelitian ini diharapkan dapat memberikan pemahaman tentang *code smells* serta cara mendeteksi *code smells* itu sendiri.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang diatas, maka didapatkan beberapa rumusan masalah, sebagai berikut:

1. Apa saja kebutuhan dalam membangun sistem untuk pendeteksian *code smells* dengan jenis *refused bequest*?
2. Bagaimana perancangan dalam membangun sistem untuk pendeteksian *code smells* dengan jenis *refused bequest*?
3. Bagaimana implementasi dalam membangun sistem untuk pendeteksian *code smells* dengan jenis *refused bequest*?
4. Bagaimana hasil pengujian dari pembangunan sistem untuk pendeteksi *code smells* dengan jenis *refused bequest*?

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini terbagi menjadi 2 bagian, antara lain adalah tujuan umum penelitian dan tujuan khusus dari penelitian. Dimana tujuan umum penelitian adalah membangun suatu sistem yang dapat melakukan pendeteksian *code smells refused bequest* di tahap perancangan pada suatu sistem yang akan diidentifikasi *code smells* tersebut. Sedangkan tujuan khusus pada penelitian ini terbagi menjadi beberapa poin, yaitu:

1. Pembangunan sistem pendeteksi *code smells refused bequest* dibangun sesuai dengan analisis kebutuhan sistem.
2. Pembangunan sistem pendeteksi *code smells refused bequest* dibangun sesuai dengan perancangan sistem.
3. Pembangunan sistem pendeteksi *code smells refused bequest* dibangun dapat memudahkan penggunanya dalam menemukan *refused bequest.*
4. Penelitian tentang sistem pendeteksi *code smells refused bequest* diharapkan memberikan kemudahan dalam memahami *refused bequest*.

1.4 Manfaat Penelitian

Pada penelitian dilakukan diuraikan beberapa manfaat yang didapat dari penelitian, sebagai berikut:

1. Pada skripsi ini memberikan pemahaman lebih dalam bagi pembaca untuk lebih memahami terkait *code smells refused bequest*.
2. Hasil dari skripsi ini dapat digunakan untuk mendeteksi suatu *code smells refused bequest* pada suatu sistem yang diidentifikasi.
3. Pada skripsi ini dapat mempermudah proses perancangan sistem untuk lebih bersih dari *refused bequest*.
4. Pada skripsi ini dapat dijadikan sebagai bahan ajar untuk melakukan penanganan terhadap *code smells refused bequest*.

1.5 Batasan Masalah Penelitian

Batasan masalah pada pembangunan sistem untuk pendeteksian *code smells refused bequest* adalah sebagai berikut:

1. Proses pembangunan sistem yang dilakukan hanya menggunakan bahasa pemrograman Java.
2. Penelitian yang dilakukan hanya terbatas kepada salah satu jenis *code smells* saja yaitu *refused bequest*.
3. Penelitian yang dilakukan adalah untuk membangun suatu sistem pendeteksi dengan berbasis *desktop application*.
4. Hasil penelitian hanya berguna untuk memudahkan proses perancangan sistem.

1.6 Sistematika Pembahasan Penelitian

BAB I PENDAHULUAN

Bab ini menguraikan tentang bab pendahuluan pada sistem pendeteksi *code smells refused bequest* seperti latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah penelitian, dan sistematika pembahasan penelitian.

BAB II LANDASAN KEPUSTAKAAN

Menguraikan tentang dasar-dasar teori dan referensi yang menjadi dasar dalam proses pembangunan sistem untuk pendeteksian *code smells refused bequest.*

BAB III METODOLOGI

Menguraikan tentang metode-metode apa saja yang digunakan dalam pembangunan sistem pendeteksi *code smells refused bequest.*

BAB IV ANALISIS KEBUTUHAN

Membahas tentang analisis kebutuhan dan melakukan pemodelan terhadap sistem yang akan dibangun sesuai dengan teknik pemodelan berorientasi obyek.

BAB V PERANCANGAN DAN IMPLEMENTASI SISTEM

Membahas tentang proses perancangan sistem serta menjelaskan hasil implementasi sistem berdasarkan rancangan sistem yang

dibuat pada pembangunan sistem untuk pendeteksian *code smells refused bequest*. Pada tahap perancangan sistem terdapat perancangan arsitektur, perancangan komponen, dan perancangan antarmuka sistem. Pada tahap implementasi sistem terdapat spesifikasi sistem, impementasi kode program berdasarkan hasil perancangan komponen, dan implementasi antarmuka sistem.

BAB VI PENGUJIAN SISTEM

Membahas tentang pengujian sistem yang dilakukan pada pembangunan sistem untuk pendeteksian *code smells refused bequest* yang meliputi pengujian unit, pengujian integrasi dan pengujian validasi pada sistem yang dibangun.

BAB VII PENUTUP

Membahas tentang kesimpulan yang dihasilkan dari penelitian yang dilakukan sesuai dengan tahapan pengembangan perangkat lunak yang meliputi tahapan analisis kebutuhan, perancangan dan implementasi sistem, dan pengujian sistem. Adapun saran berdasarkan penelitian untuk pengembangan sistem yang lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Code Smells

Code smells merupakan suatu karakteristik dari perangkat lunak yang mengindikasikan permasalahan pada kode dan desain sistem sehingga membuat perangkat lunak sulit untuk dikembangkan dan dilakukan perawatan (Fontana, 2011). Istilah *code smells* pertama kali dikemukakan oleh Fowler pada tahun 2000 yang kemudian dikembangkan oleh beberapa peneliti lainnya seperti Mantyla pada tahun 2003 sebagai *smells* yang dapat dideteksi. Sebagai dasarnya, Fowler telah mendefinisikan berbagai macam jenis *code smells* yang sering muncul pada sebuah kode program terdapat 22 jenis. Selanjutnya, berbagai penelitian mengembangkan jenis-jenis *code smells* tersebut menjadi 26 yang dapat ditemukan pada kode program dalam suatu pengembangan perangkat lunak seperti *refused bequest*, *duplicated source code, long method, god method* dan jenis-jenis *code smells* lainnya.

Terdapat suatu pendapat yang menjelaskan tentang *code smells* yang merupakan suatu *design defect* atau suatu kerusakan pada struktur kode program. Hal ini, merupakan sesuatu yang buruk sehingga dapat mengurangi kualitas perangkat lunak. *Code smells* harus diminimalisir dengan melakukan deteksi keberadaannya didalam kode program (Hanson, 2010). Untuk melakukan pendeteksian *code smells* sebenarnya dapat dilakukan secara manual oleh manusia dengan melakukan analisis terhadap struktur sistem. Namun, lebih disarankan untuk melakukan hal tersebut dengan bantuan sistem kakas bantu untuk hasil yang lebih akurat dalam menemukan *code smells* yang dicari. Beberapa kakas bantu yang telah dikembangkan dapat ditemukan untuk melakukan deteksi *code smells* seperti sebuah *plug in* untuk *framework* pada IDE Eclipse yang dikenal dengan JDeodorant. *Plug in* ini mendeteksi problem pada suatu kode program, dimana yang dideteksi adalah bahasa pemrograman Java.

2.2 Refused Bequest

Salah satu jenis *code smells* yang cukup terkenal pada permasalahan pemrograman berorientasi obyek adalah *refused bequest*. *Refused bequest* merupakan kondisi dimana konsep pewarisan tidak digunakan sebagaimana mestinya (Ligu, 2013). *Refused bequest* juga merupakan suatu bentuk penolakan oleh *subclass* kepada *superclass* dengan tidak menggunakan *method* dan data dari *superclass* (Fowler, 2000). Berdasarkan hal tersebut, apabila suatu perangkat lunak terindikasi *refused bequest* maka hal tersebut merupakan suatu permasalahan pada kualitas perangkat lunak. Beberapa pendapat dari peneliti *code smells* menerangkan bahwa permasalahan pada kualitas perangkat lunak yang diakibatkan terindikasinya *code smells* akan menyebabkan perangkat lunak tersebut sulit untuk dilakukan perawatan sistem kedepannya.

Permasalahan utama dari *code smells* ini adalah pada kode program perangkat lunak. Sehingga, banyak sekali penelitian yang telah dikembangkan untuk otomatisasi dalam melakukan pendeteksian *code smells* serta perangkat

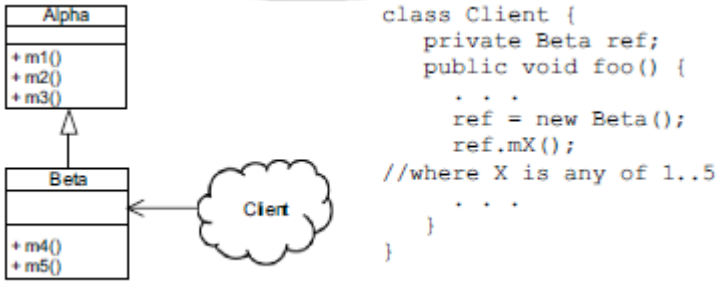
lunak yang mampu mengatasi permasalahan *code smells* ini pada tahap kode program saja. Salah satu penelitian tentang *refused bequest* adalah penelitian yang dilakukan oleh Hanson (2010) dalam jurnalnya "*Deteksi Code Smell Pada Kode Program Dalam Representasi AST Dengan Pendekatan by Rules*". Pada penelitian Hanson yang dilakukan, dihasilkan sebuah perangkat deteksi *code smells* yang menggunakan masukan berupa *Abstract Syntax Tree* (AST) dari kode program dan juga kriteria yang telah disusun dengan pendekatan *by rules*. Menurut hasil penelitian Hanson, diketahui bahwa deteksi *code smells* pada level AST dapat dilakukan dalam semua bahasa pemrograman.

2.2.1 Penelitian Deteksi *Code Smells Refused Bequest*

Pada penelitian yang dilakukan oleh Hanson adalah membangun sebuah sistem deteksi keseluruhan jenis *code smells* dengan pendekatan *Rules*. Adapun penelitian yang dilakukan oleh Elvis Ligu dalam jurnalnya "*Identification of Refused Bequest Code Smells*". Pada penelitian ini, Ligu melakukan deteksi *code smells* dengan jenis *refused bequest* saja. Dimana penelitian ini berfokus pada struktur kode program perangkat lunak serta pada uji *class* diagram sebagai skenario pada penelitiannya untuk mengidentifikasi *refused bequest*. Metodologi pada penelitian ini adalah melakukan analisis pada kode program dilihat dari struktur pewarisan serta melakukan skenario uji pada sebuah perangkat lunak untuk mendeteksi *refused bequest*. Hasil identifikasi *code smells* diurutkan kedalam kategori tertentu berdasarkan kepada jumlah method yang *di override,* pemanggilan *method* pada *superclass*, serta hasil dari skenario uji yang dilakukan. Kategori tersebut digolongkan kedalam "Termometer Smells" yang akan mempresentasikan tingkat *smells* yang ada pada perangkat lunak yang diidentifikasi.

2.2.1.1 Ide Konsep Penelitian

Ligu dalam penelitiannya melakukan deteksi *code smells* dengan jenis *refused bequest* pada suatu perangkat lunak melihat kepada kode program. Serta bagaimana pewarisan diterapkan pada sebuah perangkat lunak. Gambaran *class* diagram serta method yang ada pada *class* tersebut berdasarkan penelitian yang dilakukan Ligu, sebagai berikut:



Gambar 2.1 Class Diagram Penelitian Elvis Ligu

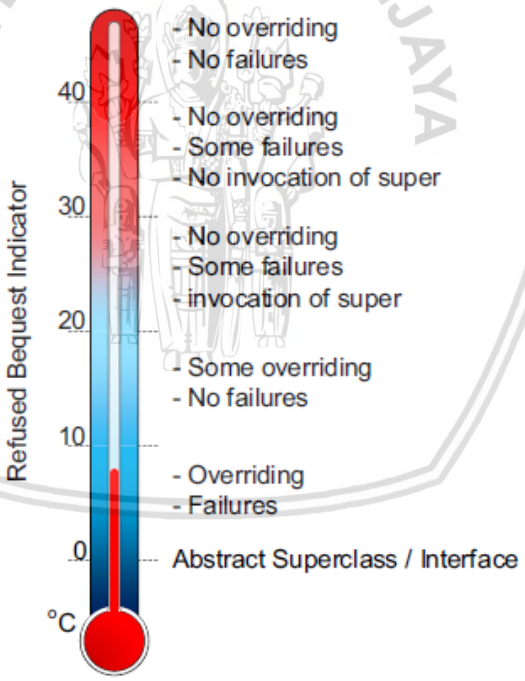
(Sumber: Identification of Refused Bequest Code Smells, 2013)

Pada class diagram tersebut, diketahui bahwa *class* Beta merupakan *subclass* dari *class* Alpha. Pada *class* Alpha terdapat *method* (`m1()`, `m2()`, dan `m3()`) sedangkan pada *class* Beta terdapat method (`m4()` dan `m5()`). Jika pada kasus

diatas *Client* hanya memanggil *method* yang ada pada *class* Beta saja, tanpa memanggil *method* yang ada pada *class* Alpha maka, seolah-olah *subclass* menolak pewarisan dari *superclass* sehingga hal tersebut apabila dilakukan deteksi maka kemunculan *refused bequest* sangatlah jelas. Satu-satunya cara untuk melakukan pemanggilan method *superclass* adalah dengan melakukan *override* pada method tersebut. Serta, *class* tersebut tidak memunculkan *refused bequest* apabila method *superclass* dipanggil oleh *Client*. Berdasarkan beberapa kondisi yang menentukan sebuah *class* dinyatakan memiliki *refused bequest.* Menurut Ligu, dapat digolongkan menjadi beberapa kondisi yang akan dimasukkan kedalam *Termometer smells*. Pada *Termometer smells* tersebut *refused bequest* digolongkan mulai dari yang terlemah hingga yang paling kuat terdeteksi.

2.2.2 Termometer Smells

Seperti yang telah disampaikan oleh Elvis Ligu (2013) pada ide konsep penelitiannya, telah diajukan sebuah pendekatan untuk mendeteksi *refused bequest* untuk mengetahui intensitas *smells* tersebut. Berdasarkan hal ini dibuatlah suatu *Temometer smells* yang dapat menentukan kategori *refused bequest* tertentu, sebagai berikut:



Gambar 2.2 Termometer Smells Elvis Ligu

(Sumber: Identification of Refused Bequest Code Smells, 2013)

Pada penelitian Ligu, Termometer smells terdapat beberapa kondisi yang mengindikasikan intensitas *smells* tersebut. Pada penelitian ini diasumsikan setiap kondisi yang ada pada Termometer smells diberikan level mulai dari level terendah smell hingga tertinggi. Adapun range level yang diberikan mulai level 0 hingga level 3 yang pada setiap penjelasan berikut diberikan level dari *smells* yang terkait. Pada level terkuat di Termometer smells dilakukan penyesuaian

pada penelitian ini, dimana untuk kondisi 4, 5 dan 6 dari Termometer smells Ligu dilakukan penyatuan kedalam 1 kondisi yaitu tidak ada overriding yang dilakukan pada subclass. Berikut merupakan penjelasan dari setiap kondisi yang ada pada Termometer smells:

2.2.2.1 Superclass abstrak atau *Interface*

Ketika pada tahap perancangan dibuat sebuah hubungan generalisasi, dan pada *parent class* merupakan sebuah *class* abstrak ataupun *interface,* seorang perancang tersebut memiliki tujuan yang jelas. Tujuannya adalah menerapkan konsep polimorfisme, dimana *class* abstrak atau interface tersebut diimplementasikan oleh *subclass*. Dalam kondisi ini, secara teoritis tidak mungkin gejala *refused bequest* dapat muncul karena tujuan utama dari generalisasi ini adalah membangun hubungan "*is-a*" antar *superclass* dan *subclass* ketika *superclass* adalah merupakan sebuah *class* abstrak atau *interface* (Ligu, 2013). Sehingga hal ini dapat diasumsikan pada skripsi ini, kondisi tersebut memasuki level 0 kemunculan *smells*.

2.2.2.2 Overriding dan *Kegagalan*

Dalam hal ini, satu atau beberapa *method* yang ada pada *superclass* telah diganti. Menurut Ligu, hal ini mengindikasikan bahwa kemunculan *refused bequest* relatif lebih kecil. Pada saat perancang mengimplementasikan ulang *method* yang telah diwariskan agar sesuai dengan fungsionalitas pada *subclass*, hal tersebut bertujuan untuk menerapkan polimorfisme (Ligu,2013). Kedua, kemunculan *error* pada sistem mengindikasikan bahwa *method* yang diwariskan oleh *superclass* kepada *subclass* dipanggil pada saat menjalankan sistem. Sehingga hal ini dapat diasumsikan pada skripsi ini, kondisi tersebut memasuki level 1 kemunculan *smells*.

2.2.2.3 Beberapa *overriding* dan Tidak ada kegagalan

Indikasi pertama kemunculan dari *refused bequest* adalah sedikitnya kegagalan sistem pada saat dilakukan pengujian diskenario uji yang dilakukan oleh Ligu. Hal ini berarti bahwa tidak ada method yang diwariskan dari *superclass* yang dipanggil pada *subclass*. Dengan kata lain, hal ini menunjukkan bahwa *subclass* menunjukkan bentuk penolakan dari pewarisan yang dilakukan. Meskipun, satu ataupun beberapa method dari *superclass* telah diganti pada *subclass*, namun kemunculan *refused bequest* mulai tampak (Ligu, 2013). Sehingga hal ini dapat diasumsikan pada skripsi ini, kondisi tersebut memasuki level 2 kemunculan *smells*.

2.2.2.4 Tidak ada *overriding*, beberapa kegagalan dan pemanggilan dari *super*

Dalam generalisasi, ketika subclass tidak mengganti method yang ada pada *superclass* setelah diwariskan maka, seorang perancang tidak menerapkan tujuan dari polimorfisme seperti dalam pola suatu perancangan *State, Strategy,* atau *Template Method* (Ligu, 2013). Adapun, ketika method yang diwariskan dipanggil saat sistem dijalankan maka, akan terdapat *error* yang berbeda dengan kondisi pada *Termometer smells* sebelumnya. Cara yang dapat dilakukan, untuk menangani permasalahan ini adalah menambahkan suatu *method* baru

kemudian menggunakan perintah *super* dalam *method* tersebut. Tujuan pemanggilan *super* pada *method* tersebut adalah menggunakan kembali method yang tidak diganti sebelumnya, kemudian merujuk kepada *superclass*. Sehingga, hal ini merupakan bentuk pewarisan walau relatif lebih lemah dan kemunculan *refused bequest* berada pada posisi yang tengah dari *Termometer smells*. Hal ini dapat diasumsikan pada skripsi ini, kondisi tersebut memasuki level 3 kemunculan *smells*.

2.2.2.5 Tidak ada *overriding,* beberapa kegagalan dan tidak ada pemanggilan dari *super*

Dalam kondisi ini, merupakan sesuatu yang berbeda dari kondisi sebelumnya dimana sedikit pemanggilan *method* yang ada pada *superclass*. Kemudian, diikuti dengan sedikitnya pula *overriding* yang dilakukan pada method yang diwariskan. Tidak ada method tambahan yang merujuk kepada *super* untuk menggunakan kembali method yang tidak diganti sebelumnya. Sehingga, hal ini mengindikasikan *refused bequest* yang sangat jelas, serta dapat diasumsikan bahwa dalam skripsi ini, kondisi tersebut masuk kedalam Level 3 dari *Termometer Smell*.

2.2.2.6 Tidak ada *overriding,* Tidak ada kegagalan

Kondisi ini merupakan kondisi terkuat dari kemunculan *refused bequest* menurut Elvis Ligu dalam penelitiannya. Pada kondisi ini, tidak ada *method* dari *superclass* yang diganti, tidak ada *method* pewarisan yang digunakan pada subclass, tidak ada suatu *method* tambahan yang mengandung pemanggilan *method* pada *superclass* (Ligu, 2013). Dengan kata lain, hal ini menjelaskan bahwa *subclass* menolak pewarisan dari *superclass* dan secara konsep generalisasi hal ini tidak menunjukkan sebuah hubungan "*is-a*". Sehingga hal ini dapat diasumsikan pada skripsi ini, kondisi tersebut memasuki level 3 kemunculan *smells*.

2.2.3 Contoh Kasus Deteksi *Code Smells*

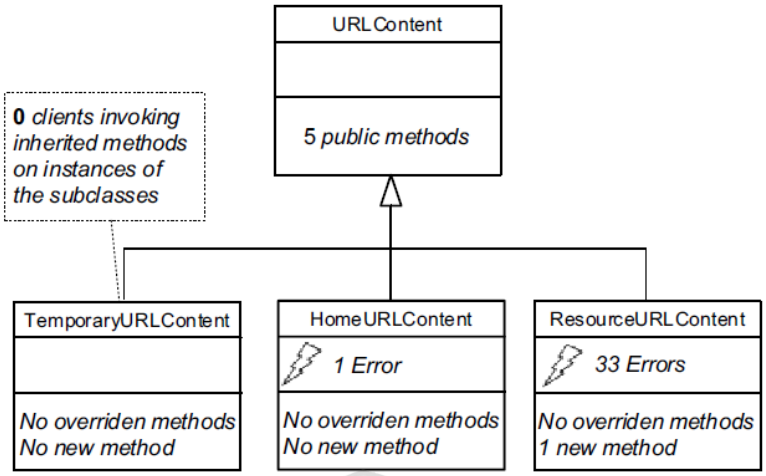
Elvis Ligu dalam penelitiannya melakukan pengujian terhadap sebuah aplikasi yang bernama SweetHome3D (v4.0) dimana aplikasi ini merupakan aplikasi *open-source* Java untuk interior design suatu rumah. *Properties* dari sistem ini akan ditunjukkan pada gambar berikut:

LOC	NUMBER OF PACKAGES	NUMBER OF CLASSES	NUMBER OF OPERATIONS	NUMBER OF INHERITANCE HIERARCHIES	NUMBER OF TEST CASES*
76730	9	460	3360	69	42

*These tests act as integration tests exercising a large portion of software features.

Gambar 2.3 Properties SweetHome3D

(Sumber: Identification of Refused Bequest Code Smells, 2013)



Gambar 2.4 Tidak ada *overriding*, Tidak ada kegagalan pada SweetHome3D (*Refused Bequest)*

(Sumber: Identification of Refused Bequest Code Smells, 2013)

Pada contoh studi kasus diatas, *class URLContent* merupakan parent class dari sistem SweetHome3D. *TemporaryURLContent* melakukan *extend* pada class tersebut. Akan tetapi, tidak ada *method* tambahan, serta tidak ada *method superclass* yang dipanggil dengan perintah *super*. Serta konsep polimorfisme tidak diterapkan semestinya, sehingga hal tersebut mengindikasikan kemunculan *refused bequest* yang kuat.

Hal yang sama dilakukan pada *class HomeURLContent* dan menunjukan sesuatu yang sedikit berbeda. Pada *class* tersebut juga tidak ada *method* tambahan, tidak ada *method* yang diganti, akan tetapi salah satu *method superclass* dipanggil melalui *subclass* pada saat menjalanakan sistem. Sehingga, hal ini menunjukkan sebuah *error* dari pemanggilan tersebut. Serta berdasarkan hasil uji yang dilakukan Ligu, hal ini mengindikasikan kemunculan *refused bequest* tetap ada meskipun method yang lainnya ditolak oleh *subclass*. Dan *refused bequest* pada class ini relatif lebih lemah dari pada *class* sebelumnya (Ligu, 2013).

Untuk *class ResourceURLContent*, terdapat setidaknya satu method yang ditambahkan, dan 33 *error* yang ditunjukkan pada hasil uji kasus studi diatas adalah penerapan seluruh *method* yang diwariskan oleh *superclass* pada *subclass* ketika sistem dijalankan. Hal ini menunjukkan bahwa *refused bequest* sulit untuk muncul, meskipun terdapat banyak sekali *error* yang ada (Ligu, 2013). Dan hal ini tidaklah melanggar konsep polimorfisme secara mendasar.

2.3 Tahapan Pengembangan Perangkat Lunak

Tahapan pengembangan sistem merupakan suatu hal yang penting dalam proses membangun suatu perangkat lunak yang sesuai dengan konsep rekayasa perangkat lunak. Pada penelitian yang akan dilakukan, terdapat beberapa pembahasan mengenai tahapan pengembangan perangkat lunak, diantaranya adalah siklus hidup pengembangan yang akan digunakan, pendekatan berorientasi obyek, serta pemodelan berorientasi obyek. Pada bagian siklus

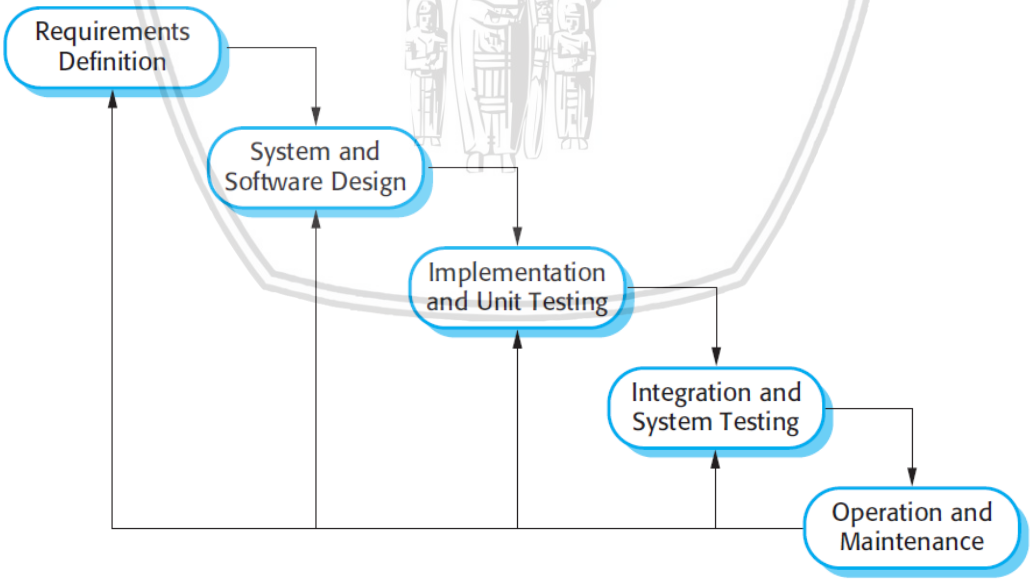
hidup pengembangan perangkat lunak akan dibahas mengenai siklus hidup pengembangan perangkat lunak yang akan digunakan yang berisi penjelasan untuk setiap tahapannya. Kemudian, pada pendekatan berorientasi obyek akan dibahas bagaimana konsep tersebut diterapkan dalam sebuah penelitian. Serta, pada bagian pemodelan berorientasi obyek akan dilakukan pembahasan mengenai pemodelan tersebut, dan bagaimana pemodelan tersebut diterapkan dalam sebuah penelitian.

2.4 Siklus Hidup Pengembangan Perangkat Lunak

Siklus hidup pengembangan perangkat lunak merupakan suatu rangkaian proses yang urut digunakan dalam membangun perangkat lunak (Kneuper, 2017). Salah satu model pengembangan perangkat lunak yang sangat terkenal adalah *waterfall.* Pada skripsi ini, akan digunakan *waterfall* sebagai siklus hidup pengembangan perangkat lunak.

2.4.1 Model Waterfall

Model pengembangan *waterfall*, biasanya disebut dengan siklus hidup pengembangan yang klasik, dimana alur dalam siklus tersebut disusun secara sistematik, serta dilakukan secara urutan (Pressman, 2010). Pada skripsi ini menggunakan siklus hidup *waterfall* dimana proses pertama yang dilakukan adalah melakukan analisis kebutuhan kemudian dari kebutuhan yang telah didapatkan dilakukan proses spesifikasi kebutuhan. Setelah itu melakukan perancangan sistem, implementasi sistem, pengujian sistem dan proses perawatan sistem. Adapun pemodel *waterfall* seperti berikut:



Gambar 2.5 Model Waterfall
(Sumber: Software Engineering 9th Edition, 2011)

2.4.2 Analisis Kebutuhan

Analisis kebutuhan sistem memiliki suatu peranan yang penting dan cukup besar dalam pembangunan suatu sistem karena proses ini merupakan titik awal yang digunakan sebagai acuan untuk langkah-langkah berikutnya (Prayitno, 2016). Pada tahap ini apabila terjadi kesalahan maka akan berdampak besar bagi tahapan selanjutnya, sehingga apabila terjadi kesalahan tersebut memungkinkan terjadi sistem yang cacat atau bahkan kegagalan sistem. Oleh karena itu, proses analisis kebutuhan sistem haruslah terencana sebaik mungkin agar dapat menghasilkan spesifikasi kebutuhan yang sesuai dengan keinginan *stakeholder* serta mampu mengatasi permasalahan yang sesuai.

2.4.3 Perancangan Sistem

Perancangan merupakan suatu tahapan selanjutnya setelah proses analisis kebutuhan sistem. Pada tahap perancangan sistem merupakan proses penerapan rancangan dari kebutuhan yang telah didapatkan kedalam suatu perangkat keras atau perangkat lunak secara keseluruhan (Sommerville, 2011). Perancangan sistem mengidentifikasi dan menjelaskan secara dasar bagaimana perangkat lunak secara abstrak serta hubungan apa saja yang ada pada suatu perangkat lunak tersebut. Pada skripsi ini, tahapan perancangan menjadi obyek penting dalam melakukan proses deteksi *code smells* dengan jenis *refused bequest*.

2.4.4 Implementasi Sistem

Implementasi sistem merupakan suatu tahapan inti dari proses pembangunan suatu sistem, dimana pada fase ini proses menerapkan hasil rancangan sistem berdasarkan kebutuhan yang telah didapatkan akan dilakukan (Sommerville, 2011). Implementasi yang dilakukan dapat berupa proses pembangunan sistem dengan kode program atau produk perangkat lunak yang akan dirilis kepada *stakeholder*. Pada bagian ini pula hasil perancangan akan diterapkan kedalam suatu unit program (Sommerville, 2011). Dan setiap fungsional sistem yang telah dirancang akan diterapkan pada bagian implementasi.

2.4.5 Pengujian Sistem

Pengujian sistem merupakan suatu tahapan yang penting untuk melakukan uji pada setiap fungsionalitas yang telah dibangun sesuai dengan kebutuhan sistem (Sommervile, 2011). Pengujian yang dilakukan pada fase ini seperti pengujian *black box* yang menguji fungsionalitas sistem tampak luar saja, sedangkan untuk pengujian *white box* adalah melakukan pengujian pada struktur inti sistem yang telah dibangun. Setelah dilakukan pengujian, perangkat lunak dapat dirilis kepada pengguna.

2.4.6 Perawatan Sistem

Fase ini merupakan fase terlama dari sebuah siklus hidup perangkat lunak. Perangkat lunak telah terpasang dan digunakan pengguna dalam waktu yang cukup lama. Fase perawatan akan memperbaiki *error* yang muncul ketika

perangkat lunak dijalankan yang tidak dicakup pada fase pengujian sistem (Sommervile, 2011). Fase perawatan sistem juga dapat mengembangkan secara lebih fungsionalitas sistem sebagai kebutuhan baru yang mengikuti perkembangan itu sendiri.

2.5 Pendekatan Berorientasi Obyek

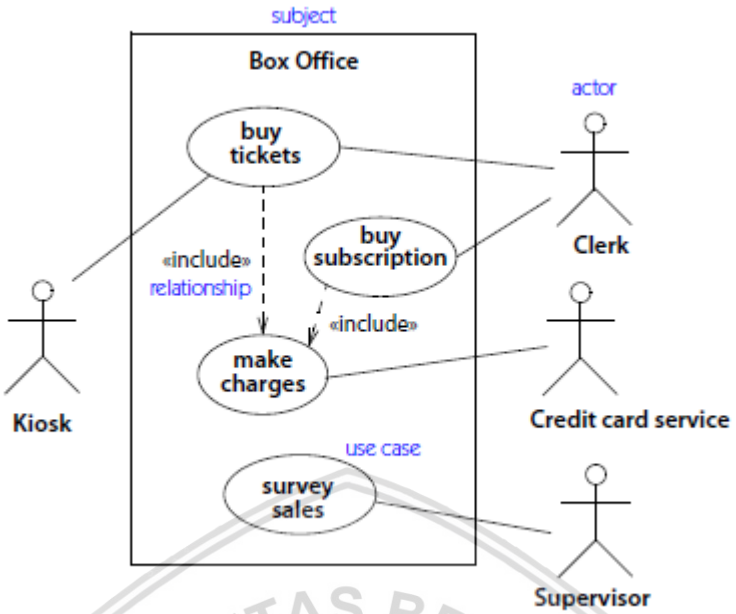
Bahasa pemrograman berorientasi obyek pertama secara umum diakui oleh Simula-67, dikembangkan di Norway pada tahun 1967 (Rumbaugh, 2004). Pendekatan berorientasi obyek ini biasa disebut dengan OO (Object Oriented) merupakan suatu pendekatan dalam pengembangan perangkat lunak. Konsep berorientasi obyek ini dibuat berdasarkan interaksi antar obyek yang mengatur *state* nya dan operasi apa yang ada pada *state* tersebut (Sommervile, 2011). Pendekatan berorientasi obyek berguna merancang suatu *class* obyek dan bagaimana relasi antar *class* itu terjadi. Pendekatan ini lebih memudahkan untuk proses pergantian fungsional sistem daripada pendekatan structural. Mengganti implementasi suatu obyek atau menambahkan suatu implementasi pada obyek tersebut tidak akan mempengaruhi obyek lainnya. Pendekatan ini tentu saja lebih mudah dalam memberikan pemahaman, serta penanganan tertentu pada suatu permasalahan, serta memberikan kemudahaan dalam perancangan perangkat lunak.

2.6 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan suatu pemodelan yang bertujuan khusus untuk melakukan visualisasi yang digunakan untuk melakukan spesifikasi, membangun, serta artifak dari proses dokumentasi suatu sistem perangkat lunak (Rumbaugh, 2004). UML dikembangkan dengan tujuan untuk melakukan penyederhanaan dari metode pemodelan object oriented yang telah diketahui. Beberapa jenis UML yang akan digunakan dalam skripsi ini, antara lain:

2.6.1 Use Case Diagram

Use case diagram merupakan suatu bentuk pemodelan untuk menggambarkan perilaku sistem yang akan dibuat, mengetahui fungsi-fungsi apa saja yang ada dalam sebuat sistem dan siapa yang dapat melakukan fungsi tersebut (Rumbaugh, 2004). Diagram ini juga dapat didefinisikan untuk menggambarkan interaksi antar sistem dengan lingungannya, serta menjelaskan interaksi antar sistem tersebut dengan actor eksternal sistem. Digram *use case* dibuat berdasarkan kebutuhan fungsional yang telah dilakukan spesifikasi, tujuannya adalah agar pembaca dari diagram ini dapat melihat gambaran serta memahami fungsionalitas dari sistem yang akan dibangun.



Gambar 2.6 Use Case Diagram

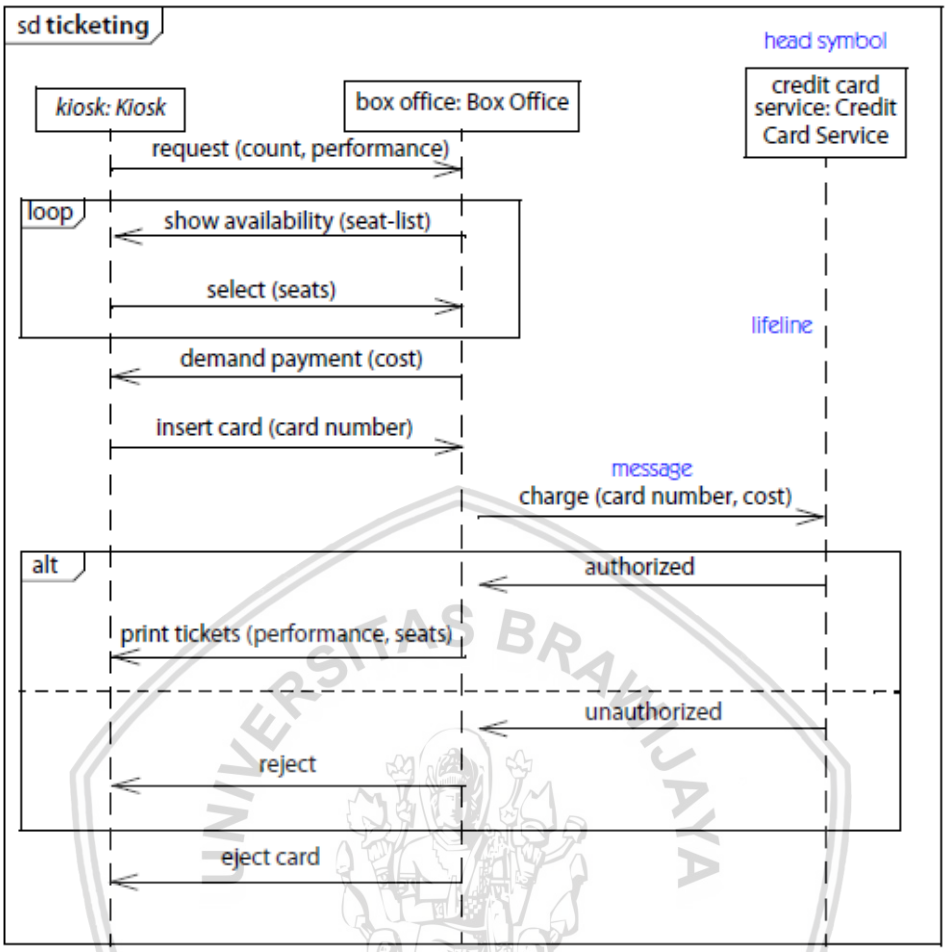
(Sumber: *Unified Modeling Language Reference Manual 2nd Edition,* 2010).

2.6.2 Use Case Scenario

Use case scenario merupakan salah satu bentuk pemodelan berorientasi obyek dimana fungsi scenario adalah menjelaskan secara detail bagaimana use case bekerja (Rumbaugh, 2004). Pada use case scenario terdapat aktor yang menggunakan *use case* tersebut, *main flow* dari *use case* tersebut, *alternative flow* dari *use case* tersebut, kondisi *pre* (sebelum) dan kondisi *post* (setelah).

2.6.3 Sequence Diagram

Sequence diagram merupakan diagram alir berdasarkan kepada satuan waktu tertentu, yang menggambarkan perilaku secara urut dari sebuah *use case*. Ketika perilaku dari *use case* di implementasikan, setiap pesan yang terdapat pada sequence diagram merupakan sebuah *operasi* dalam suatu *class* atau sebuah *event* (Rumbaugh, 2004). Berikut merupakan contoh *sequence* diagram, antara lain:

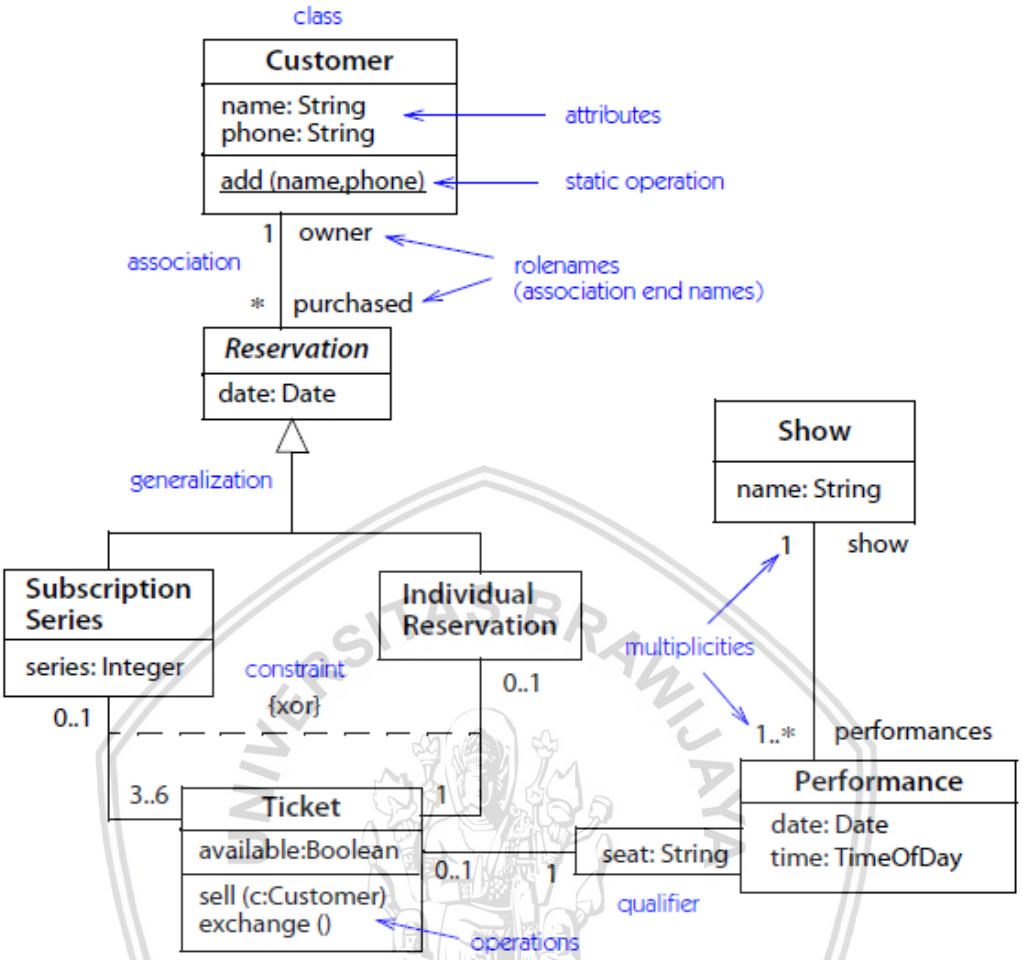


Gambar 2.7 Sequence Diagram

(Sumber: *Unified Modeling Language Reference Manual 2nd Edition,* 2010).

2.6.4 Class Diagram

Class diagram merupakan pemodelan statis untuk sebuah sistem, pada *class* diagram terdapat hubungan antar *class* dan turunannya (Rumbaugh, 2004). Pada setiap hubungan antar *class* memiliki jenis relasi tertentu berdasarkan kepada aturan penulisan *class* diagram. Berikut merupakan contoh *class* diagram, antara lain:



Gambar 2.8 *Class* diagram

(Sumber: *Unified Modeling Language Reference Manual 2nd Edition,* 2010).

2.7 Teori Pengujian

Pengujian perangkat lunak merupakan sebuah proses dari eksekusi program yang memiliki tujuan utama adalah menemukan error, kegagalan, yang ditujukan untuk secara intens menghilangkan kecacatan pada suatu perangkat lunak (Runeson, 2011). Software Testing dapat berguna untuk menguji apakah perangkat lunak yang telah dibuat memenuhi standar pengujian yang ada, guna meningkatkan kualitas dari perangkat lunak tersebut. Terdapat beberapa teknik dalam melakukan pengujian perangkat lunak, sebagai berikut:

2.7.1 Pengujian *White Box*

Pengujian *White Box* adalah suatu pengujian yang berbeda dari pengujian Black Box dimana pada pengujian ini, yang diuji adalah Struktur dari program tersebut (Runeson, 2011). Pengujian yang dilakukan adalah dengan menguji *source code* dari program, logika program dimana lebih fokus ke inti dari program tersebut. Pada dasar nya proses ini tetap melakukan input kepada program, dan setelah input diberikan maka dilakukan perhitungan dari proses logika program sehingga program dapat menampilkan output berdasarkan

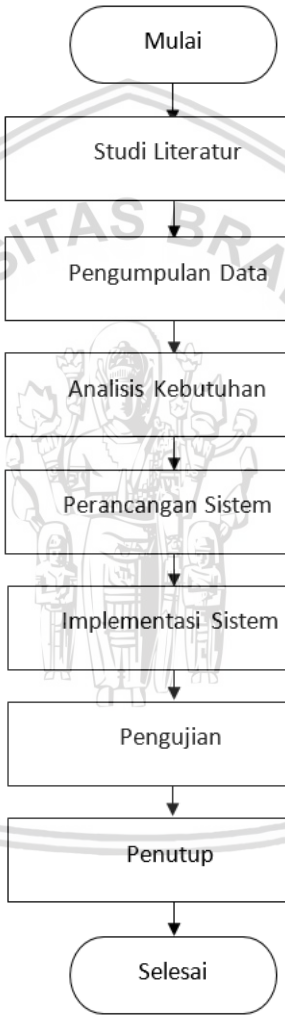
source code tersebut. Pada White Box terdapat strategi pengujian seperti *Unit Test*, *Integration* atau *System level*. Pada White Box menemukan error lebih mudah sebelum program diluncurkan sehingga terjadi kegagalan atau beberapa masalah nantinya.

2.7.2 Pengujian *Black Box*

Black Box merupakan jenis pengujian berdasarkan hasil dari beberapa input yang diberikan (uji kasus) tanpa analisis kode dari program (Gaur, 2016). Black box testing bisa juga dikategorikan sebagai uji fungsionalitas system dimana pada tahap ini kita menguji berdasarkan input yang diberikan dan output dari program akan langsung terlihat. Tujuan utama dari pengujian *Black Box* adalah menjamin apakah input yang diberikan telah sesuai dengan outputnya dan apakah sesuai dengan fungsionalitas dari sistem dan apakah itu merupakan kebutuhan dari sistem.

BAB 3 METODOLOGI PENELITIAN

Metodologi penelitian pada skripsi ini adalah suatu bab yang berguna membahas metode-metode apa saja yang akan digunakan dalam pembangunan sistem untuk pendeteksian *code smells refused bequest*. Metodologi penelitian ini terdiri dari beberapa tahapan, dimana mulai dari studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian sistem dan tahap akhir yaitu penutup yang berisi kesimpulan dan saran penelitian.



Gambar 3.1 Diagram Alir Metode Penelitian

3.1 Studi Literatur

Pada tahap studi literatur merupakan tahap dimana peneliti melakukan pembelajaran terkait ilmu-ilmu pengetahuan yang terkait dengan topik penelitian. Studi literatur memiliki tujuan untuk mendukung proses-proses penelitian dengan dasar teori yang dapat digunakan untuk tujuan pemecahan masalah penelitian agar proses penelitian dapat dilakukan dengan baik dan

benar dengan suatu dasar yang kuat. Dasar teori dari studi literatur didapatkan peneliti melalui jurnal-jurnal internasional, buku, artikel serta beberapa penelitian sebelumnya yang dapat digunakan untuk panduan terhadap penelitian ini. Adapun beberapa ilmu pengetahuan yang terkait dengan studi literatur yang akan dilakukan sebagai berikut:

1. Rekayasa perangkat lunak.
2. Kualitas dan keandalan perangkat lunak.
3. Code smells.
4. Membaca jurnal tentang *refused bequest*, seperti:
   a. *Identification of Refused Bequest Code Smells* (Ligu, 2013).
   b. *Deteksi Code Smell Pada Kode Program Dalam Representasi Ast Dengan Pendekatan by Rules* (Hanson, 2010).
5. Pemrograman Java.

3.2 Pengumpulan Data

Pada bagian ini merupakan proses penelitian untuk menentukan masukan pada sistem. Dalam hal ini adalah sebuah *class diagram* yang merupakan sebuah aplikasi *open source* yaitu SweetHome3d. Aplikasi ini merupakan aplikasi *Java* untuk desain interior rumah. Dokumentasi dan *source code* dari aplikasi ini dapat diunduh pada contoh-contoh aplikasi *Java* yang ada diinternet. Pada penelitian ini akan diuji *class diagram* dari aplikasi tersebut sebagai masukan dari sistem pendeteksian. Pada data masukan, *class diagram* dibuat pada aplikasi UML *Creator Visual Paradigm* dan kemudian dikonversi kedalam bahasa xml.

3.3 Analisis Kebutuhan

Pada tahap analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan umum yang kemudian akan di spesifikkan kebutuhan tersebut dan diperlukan dalam proses perancangan sistem. Dalam pembangunan sistem ini menggunakan konsep *object oriented*, sehingga tahap-tahap pemodelan yang dilakukan dimulai dari penggunaan diagram-diagram UML (*Unified Modeling Language*) seperti *use case* diagram dan *use case scenario*. Pada pembuatan use case diagram didasarkan untuk mendeskripsikan kebutuhan - kebutuhan dan fungsionalitas perangkat lunak dari perspektif end-user. Tahap analisis kebutuhan dilakukan dengan melakukan identifikasi semua kebutuhan (requirements) sistem yang kemudian dimodelkan kedalam diagram *use case*.

3.4 Perancangan Sistem

Perancangan sistem yang dilakukan pada penelitian ini merupakan tahap proses pembangunan sistem lanjutan dari analisis kebutuhan. Pada perancangan sistem penelitian ini adalah membuat perancangan sequence diagram, *class* diagram, komponen, serta antarmuka sistem. Tahap perancangan sistem merupakan acuan untuk melakukan implementasi sistem kemudian diikuti dengan pengujian sistem.

3.5 Implementasi Sistem

Pada tahap implementasi sistem merupakan tahap pada penelitian untuk membangun sistem berdasarkan hasil rancangan yang telah dibuat. Tahap ini pula peneliti membuat sistem secara keseluruhan berdasarkan kepada kebutuhan sistem yang telah didefinisikan. Sistem yang dibangun berbasis *desktop application* dengan menggunakan bahasa pemrograman Java. Implementasi sistem yang akan dilakukan dimulai dengan membangun antarmuka sistem terlebih dahulu, kemudian memberikan fungsionalitas sistem sesuai dengan kebutuhan yang telah ditentukan.

3.6 Pengujian Sistem

Pada tahap ini menjelaskan tentang pengujian yang dilakukan terhadap sistem yang telah dibangun berdasarkan kebutuhan/permasalahan yang telah didapatkan pada tahap awal. Pengujian yang dilakukan mengikuti strategi pengujian yang ada pada dokumen uji perangkat lunak yang merupakan dokumentasi untuk proses pengujian. Diantaranya, terdapat pengujian *white box* yang menguji bagian inti dari sistem serta pengujian *black box*, dimana menguji setiap fungsionalitas dari sistem.

3.7 Penutup

Tahapan akhir yaitu penutup yang berisi kesimpulan dan saran yang akan diambil ketika semua tahapan pada penelitian terselesaikan. Kesimpulan diambil dari hasil akhir pemodelan dan pengujian sistem yang dapat menjawab perumusan masalah yang telah dirumuskan. Saran yang diberikan berfungsi untuk mengevaluasi jika terjadinya kesalahan dan memberikan masukan kepada penulis untuk penelitian selanjutnya.

BAB 4 ANALISIS KEBUTUHAN SISTEM

4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan merupakan tahapan awal dalam melakukan pengembangan sistem, pada tahap ini merupakan proses penggalian kebutuhan yang akan menjadi kebutuhan apa saja yang harus ada pada sistem. Tujuan dari adanya analisis kebutuhan adalah untuk pencapaian kebutuhan sistem yang mencukupi dan mendukung proses pengembangan sistem. Pada penelitian ini, kebutuhan didapatkan dari melakukan analisis penelitian Elvis Ligu (2013) tentang identifikasi *code smells* berjenis *refused bequest*.

Pada analisis kebutuhan juga menjelaskan siapa saja yang berperan dalam penggunaan sistem. Adapun pada analisis kebutuhan ini juga terdapat pemodelan-pemodelan kebutuhan dengan pendekatan berorientasi obyek seperti diagram *use case*, *use case scenario* yang berguna untuk melakukan penjelasan lebih detail dari diagram *use case* yang telah dibuat.

4.2 Gambaran Umum Sistem

Pembangunan sistem untuk mendeteksi *code smells* berjenis *refused bequest* ini merupakan sistem yang bertujuan untuk memudahkan penggunanya dalam mendeteksi *refused bequest* pada tahap perancangan secara otomatis. Proses awal untuk mendeteksi *refused bequest* adalah dengan melakukan *parsing class* diagram dari salah satu aplikasi UML *Creator Visual Paradigm* dengan format .vpp kedalam bahasa XML. Setelah dilakukan *parsing*, file.XML dimasukkan kedalam sistem deteksi lalu selanjutnya dianalisis oleh sistem, sehingga sistem dapat menemukan *refused bequest* yang terdapat dalam *class* diagram tersebut. Proses analisis yang dilakukan oleh sistem deteksi didapatkan dari jurnal Elvis Ligu dengan judul "*Identification of Refused Bequest Code smells*" yang merupakan jurnal utama dari penelitian ini sebagai acuan dalam melakukan penggalian kebutuhan dan penelitian itu sendiri. Refused bequest yang dideteksi digolongkan kedalam 6 kondisi berdasarkan *termometer smells*.

4.3 Identifikasi Aktor

Pada tahap ini dilakukan Identifikasi Aktor yang bertujuan untuk menjelaskan aktor yang nantinya akan melakukan interaksi pada sistem. Aktor tersebut akan dijelaskan secara detail pada tabel Identifikasi Aktor, sebagai berikut:

Tabel 4.1 Identifikasi Aktor

No	Aktor	Deskripsi
1	**User**	*User* merupakan pengguna dalam sistem pendeteksi *code smells refused bequest.*

4.4 Kebutuhan Fungsional

Kebutuhan fungsional sistem merupakan suatu fungsionalitas atau layanan yang harus ada pada sistem. Kebutuhan fungsional didapatkan dari hasil penggalian kebutuhan yang kemudian dilakukan analisis kebutuhan serta dispesifikkan menjadi fungsi utama pada sistem yang akan dibangun. Pada skripsi ini, kebutuhan didapatkan dari jurnal utama oleh Elvis Ligu dan dilakukan beberapa pengembangan, agar kebutuhan utama sistem jelas dan tujuan ketercapaian pembangunan sistem dapat terpenuhi. Kebutuhan fungsional sistem pendeteksi *code smells* berjenis *refused bequest* dapat dilihat pada Tabel 4.2 berikut. Adapun untuk setiap kebutuhan diberikan penomoran untuk mengidentifikasi kebutuhan yang ada. Pada skripsi ini, aturan penomoran pada kebutuhan fungsional adalah CSRB-F-X dimana CSRB merupakan singkatan dari *Code Smells Refused Bequest*, F merupakan inisial untuk kebutuhan fungsional, serta X merupakan nomor kebutuhan fungsional.

a. User

Tabel 4.2 Kebutuhan Fungsional

No	Kode	Deskripsi Kebutuhan	Use Case
1	CSRB-F-001	Sistem harus mampu memilih masukan sistem oleh *user* berupa file XML dari *class diagram* yang akan di identifikasi.	Memilih File XML
	CSRB-F-001-01	Sistem harus mampu menampilkan *open dialog* yang berguna untuk memilih file yang akan dimasukkan kedalam sistem setelah menekan tombol *choose file* serta menyimpan masukan file dengan ekstensi xml.	
	CSRB-F-001-02	Sistem hanya dapat menerima masukan file dengan ekstensi xml saja.	
	CSRB-F-001-03	Sistem dapat memberikan notifikasi apabila file terpilih yang dimasukkan telah sesuai.	
2	CSRB-F-002	Sistem harus mampu diberi masukan file xml yang telah dipilih kedalam sistem.	Memasukkan File XML

	CSRB-F-002-01	Sistem menyediakan tombol *submit* ketika file telah dipilih dalam sistem.	
	CSRB-F-002-02	Sistem memberikan notifikasi apabila file berhasil dimasukkan atau belum ada file yang dimasukkan.	
3	CSRB-F-003	Sistem harus mampu menghapus file xml yang telah dipilih kedalam sistem.	Menghapus File XML
	CSRB-F-003-01	Sistem menyediakan tombol *remove* ketika file telah dipilih dalam sistem.	
	CSRB-F-003-02	Sistem memberikan notifikasi apabila file berhasil dihapus atau belum ada file yang dimasukkan.	
4	CSRB-F-004	Sistem harus mampu mendeteksi file xml yang dimasukkan kedalam sistem.	Mendeteksi File XML
	CSRB-F-004-01	Sistem harus mampu melakukan proses pendeteksian *refused bequest* pada *class diagram* yang dimasukkan kedalam sistem saat *user* menekan tombol *detect*.	
	CSRB-F-004-02	Sistem memberikan notifikasi saat berhasil melakukan deteksi sebuah file xml yang dimasukkan.	
5	CSRB-F-005	Sistem harus mampu menampilkan sebuah ringkasan hasil deteksi ketika berhasil melakukan deteksi file.	Melihat Ringkasan Hasil Deteksi File XML

	CSRB-F-005-01	Sistem menyediakan tombol *view summary* untuk melihat ringkasan hasil deteksi file.	
	CSRB-F-005-02	Sistem menampilkan ringkasan hasil deteksi seperti nama file yang dideteksi, jumlah *class* yang dideteksi, serta jumlah hubungan realisasi file yang dideteksi dan jumlah *interface* yang dideteksi.	
6	CSRB-F-006	Sistem harus mampu melakukan klasifikasi hasil deteksi file kedalam kategori *smells*.	Melihat Kategori File XML
	CSRB-F-006-01	Sistem menampilkan hasil deteksi kedalam kategori berdasarkan level pada *thermometer smells.*	
	CSRB-F-006-02	Sistem menyediakan tombol untuk *view category* dimana pada tombol tersebut ketika dipilih pengguna akan menampilkan halaman yang menampilkan hasil deteksi kedalam *thermometer smells.*	
	CSRB-F-006-03	Pada halaman *view category* terdapat beberapa hasil deteksi yang ditampilkan berdasarkan jumlah realisasi.	

4.5 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan batasan-batasan dari fungsionalitas utama sistem atau layanan sistem. Kebutuhan non-fungsional pada skripsi ini dapat dilihat pada Tabel 4.3 berikut. Adapun aturan penomoran yang sama halnya dengan aturan penomoran pada kebutuhan fungsional antara lain adalah CSRB-NF-X dimana CSRB merupakan singkatan dari *Code Smells Refused Bequest*, NF merupakan inisial untuk kebutuhan non-fungsional, serta X merupakan nomor dari kebutuhan non-fungsional.

Tabel 4.3 Kebutuhan Non-Fungsional

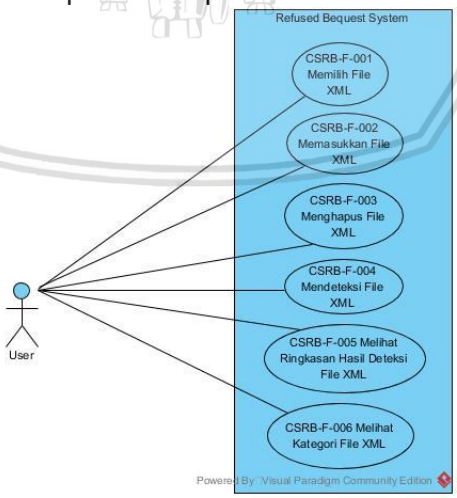
No	Kode	Kebutuhan
1	CSRB-NF-001	Sistem dapat melakukan proses deteksi refused bequest dengan batas waktu respon sistem dibawah 10 detik.

4.6 Pemodelan Kebutuhan

Pemodelan kebutuhan merupakan suatu proses membangun deskripsi sistem secara umum dengan model dari perancangan sistem. Pemodelan kebutuhan menjadi dasar bagi perancangan perangkat lunak serta menjadi referensi dalam proses validasi kebutuhan. Pada pemodelan kebutuhan penelitian ini, mengikuti konsep *object oriented* dimana proses pemodelan dimulai dengan membangun *use case diagram* dan *use case scenario*.

4.6.1 Use Case Diagram

Use case diagram merupakan suatu bentuk pemodelan untuk menggambarkan perilaku sistem yang akan dibuat, mengetahui fungsi-fungsi apa saja yang ada dalam sebuat sistem dan siapa yang dapat melakukan fungsi tersebut (Rumbaugh, 2004). Diagram ini juga dapat didefinisikan untuk menggambarkan interaksi antar sistem dengan lingungannya, serta menjelaskan interaksi antar sistem tersebut dengan actor eksternal sistem. Digram *use case* dibuat berdasarkan kebutuhan fungsional yang telah dilakukan spesifikasi, tujuannya adalah agar pembaca dari diagram ini dapat melihat gambaran serta memahami fungsionalitas dari sistem yang akan dibangun. *Use case diagram* yang dibuat pada skripsi ini dapat dilihat pada Gambar 4.1 berikut:



Gambar 4.1 *Use Case Diagram*

4.6.2 Use Case Scenario

Use case scenario merupakan salah satu bentuk pemodelan berorientasi obyek dimana fungsi scenario adalah menjelaskan secara detail bagaimana use

case bekerja (Rumbaugh, 2004). Pada use case scenario terdapat aktor yang menggunakan *use case* tersebut, *main flow* dari *use case* tersebut, *alternative flow* dari *use case* tersebut, kondisi *pre* (sebelum) dan kondisi *post* (setelah). *Use case scenario* yang dibuat pada skripsi ini, dapat dilihat pada Tabel 4.4 hingga Tabel 4.9 berikut:

Tabel 4.4 *Use Case Scenario* Memilih File XML

Nama Use Case	Memilih File XML
Kode Kebutuhan	CSRB-F-001
Actor	User
Objective	*Use case* ini berfungsi untuk memilih masukan sistem berupa file dengan ekstensi xml oleh aktor.
Pre-Condition	Aktor berada pada halaman Home.
Main Flow	1. Aktor memilih menu Analyze pada form. 2. Aktor menekan tombol "Choose File" pada form yang ada. 3. Sistem menampilkan menu *Open File Dialog*. 4. Aktor memilih file.xml dan menekan tombol "Open". 5. Sistem menampilkan nama file beserta ekstensi file pada *text field* yang ada di form tersebut.
Alternative Flow	1. Aktor tidak memilih sebuah file dan menekan tombol "Cancel" 2. Sistem menampilkan form serta notifikasi pada text field "No File chosen!".
Post-Condition	Aktor dapat memasukkan file kedalam sistem.

Tabel 4.5 *Use Case Scenario* Memasukkan File XML

Nama Use Case	Memasukkan File XML
Kode Kebutuhan	CSRB-F-002
Actor	User
Objective	*Use case* ini berguna untuk memasukkan file dengan ekstensi xml berdasarkan file yang dipilih aktor.
Pre-Condition	Aktor telah memilih file.xml untuk dimasukkan kedalam sistem dan telah berada pada halaman

	Home.
Main Flow	1. Aktor menekan tombol *Submit* pada form. 2. Sistem melakukan submit file dengan ekstensi XML kedalam sistem. 3. Sistem menampilkan notifikasi "File Submitted!"
Alternative Flow	1. Aktor menekan tombol *Submit* pada sistem yang belum dipilih sebuah file xml. 2. Sistem memberikan notifikasi "Please Add a File!"
Post-Condition	Aktor dapat melakukan submit file xml kedalam sistem.

Tabel 4.6 *Use Case Scenario* Menghapus File XML

Nama Use Case	Menghapus File XML
Kode Kebutuhan	CSRB-F-003
Actor	User
Objective	*Use case* ini berguna untuk menghapus file.xml yang telah dipilih aktor.
Pre-Condition	Aktor telah memilih file.xml untuk dimasukkan kedalam sistem dan telah berada pada halaman Home.
Main Flow	1. Aktor menekan tombol *Remove* pada form. 2. Sistem menghapus file.xml yang telah dimasukkan kedalam sistem dan memberikan notifikasi pada text field bahwa "File has been Removed!".
Alternative Flow	1. Aktor menekan tombol *Remove* pada sistem yang belum dimasukkan sebuah file xml. 2. Sistem memberikan notifikasi "Please Add a File!"
Post-Condition	Aktor dapat menghapus file dalam sistem.

Tabel 4.7 *Use Case Scenario* Mendeteksi File XML

Nama Use Case	Mendeteksi File XML

Kode Kebutuhan	CSRB-F-004
Actor	User
Objective	*Use case* ini berguna untuk mendeteksi file dengan ekstensi XML.
Pre-Condition	Aktor berada pada halaman Home dan telah memasukkan file.xml kedalam sistem.
Main Flow	1. Aktor menekan tombol *Detect* pada form. 2. Sistem melakukan proses deteksi *refused bequest* pada masukkan file xml yang diberikan. 3. Sistem memberikan informasi berupa notifikasi "File has been checked!" bahwa file yang dimasukkan telah berhasil dideteksi.
Alternative Flow	1. Aktor menekan tombol *Detect* saat file belum disubmit atau dimasukkan kedalam sistem. 2. Sistem menampilkan notifikasi "Please add a file".
Post-Condition	Aktor dapat melakukan deteksi file pada sistem.

Tabel 4.8 *Use Case Scenario* Melihat Ringkasan Hasil Deteksi File XML

Nama Use Case	Melihat Ringkasan Hasil Deteksi File XML
Kode Kebutuhan	CSRB-F-005
Actor	User
Objective	*Use case* ini berguna untuk melihat ringkasan hasil deteksi file xml.
Pre-Condition	Aktor telah berhasil melakukan deteksi file dan berada pada menu *Result*.
Main Flow	1. Aktor menekan tombol *View Summary*. 2. Sistem menampilkan hasil deteksi yang dilakukan pada menu *Analyze* kedalam sebuah halaman.
Alternative Flow	1. Aktor menekan tombol *View Summary* saat file belum dideteksi pada menu *Analyze* 2. Sistem menampilkan notifikasi untuk

	melakukan deteksi file terlebih dahulu seperti "Please check a file first!"
Post-Condition	Aktor dapat melihat ringkasan hasil deteksi file.

Tabel 4.9 *Use Case Scenario* Melihat Kategori File XML

Nama Use Case	Melihat Kategori File XML
Kode Kebutuhan	CSRB-F-006
Actor	User
Objective	*Use case* ini berguna untuk melihat kategori file kedalam *termometer smells*.
Pre-Condition	Aktor telah berhasil melakukan deteksi file dan berada pada menu *Result*.
Main Flow	1. Aktor menekan tombol *View Category*. 2. Sistem menampilkan hasil deteksi yang dilakukan pada menu *Analyze* kedalam sebuah halaman.
Alternative Flow	1. Aktor menekan tombol *View Category* saat file belum dideteksi pada menu *Analyze* 2. Sistem menampilkan notifikasi untuk melakukan deteksi file terlebih dahulu seperti "Please check a file first!"
Post-Condition	Aktor dapat melihat kategori file hasil deteksi kedalam *termometer smells*.

BAB 5 PERANCANGAN DAN IMPLEMENTASI SISTEM

5.1 Perancangan Sistem

Perancangan sistem dilakukan setelah melakukan proses analisis kebutuhan, dimana proses ini merupakan tahapan lanjutan dari analisis kebutuhan dalam proses pengembangan perangkat lunak. Perancangan sistem yang dilakukan dibagi menjadi tiga tahapan yang meliputi perancangan arsistektur, perancangan komponen, serta perancangan antarmuka. Perancangan digunakan untuk tahap implementasi sistem, perancangan merupakan acuan dalam proses pembangunan sistem untuk pendeteksian *code smells refused bequest*.

5.1.1 Perancangan Arsitektur

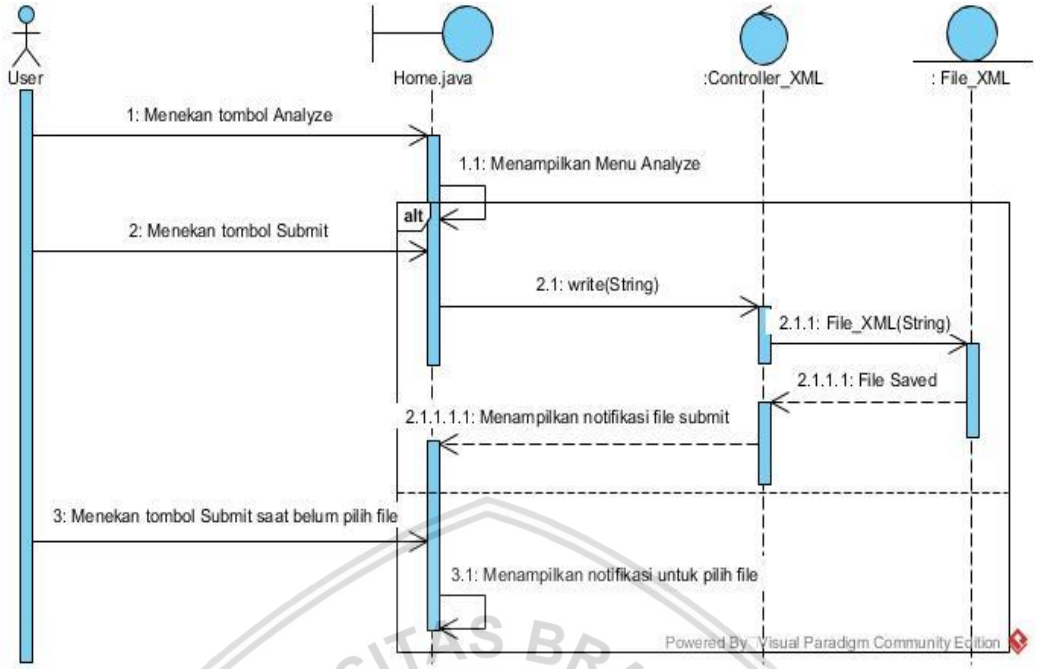
Perancangan arsitektur merupakan tahapan pengembangan untuk menjelaskan secara detail tentang *sequence diagram* dan *class diagram* pada sistem. Pada perancangan arsitektur ini akan dijelaskan *sequence diagram* yang menjelaskan alur interaksi antar objek dalam sistem, akan dijelaskan kedalam tiga diagram *sequence* sebagai fitur utama sistem. Ketiga diagram tersebut yaitu, memasukkan file xml, mendeteksi file xml, serta melihat kategori file xml. Adapun *class diagram* akan dijelaskan mengenai gambaran *class-class* beserta hubungannya sebagai penyusun sistem yang saling berinteraksi.

5.1.1.1 Sequence Diagram

Pada bagian ini akan dijelaskan ketiga *sequence diagram* yang menjadi fitur utama sistem dalam melakukan proses pendeteksi *code smells refused bequest* yaitu, memasukkan file xml, mendeteksi file xml, serta melihat kategori file xml.

5.1.1.1.1 Sequence Diagram Memasukkan File XML

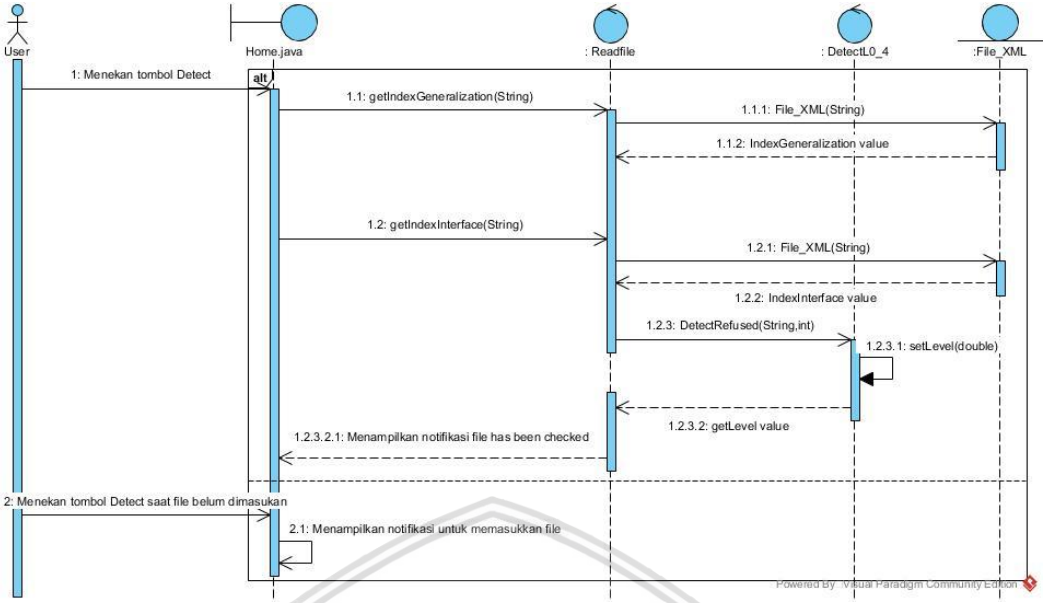
Pada Gambar 5.1 merupakan *sequence diagram* untuk memasukkan file xml yang telah dipilih *user* untuk dilakukan deteksi *code smells refused bequest*. Langkah pertama yang dilakukan *user* adalah dengan menekan tombol *analyze* pada tampilan sistem yang digambarkan dalam *sequence diagram* kedalam obyek *boundary* yaitu Home.java dan sistem akan menampilkan menu *analyze*. Selanjutnya, ketika *user* menekan tombol *submit* maka operasi yang ada pada obyek *controller* dijalankan, pada *sequence diagram* ini Controller_XML adalah obyek *controller*. Adapun obyek *entity* pada *sequence diagram* ini adalah File_XML.



Gambar 5.1 *Sequence Diagram* Memasukkan File XML

5.1.1.1.2 Sequence Diagram Mendeteksi File XML

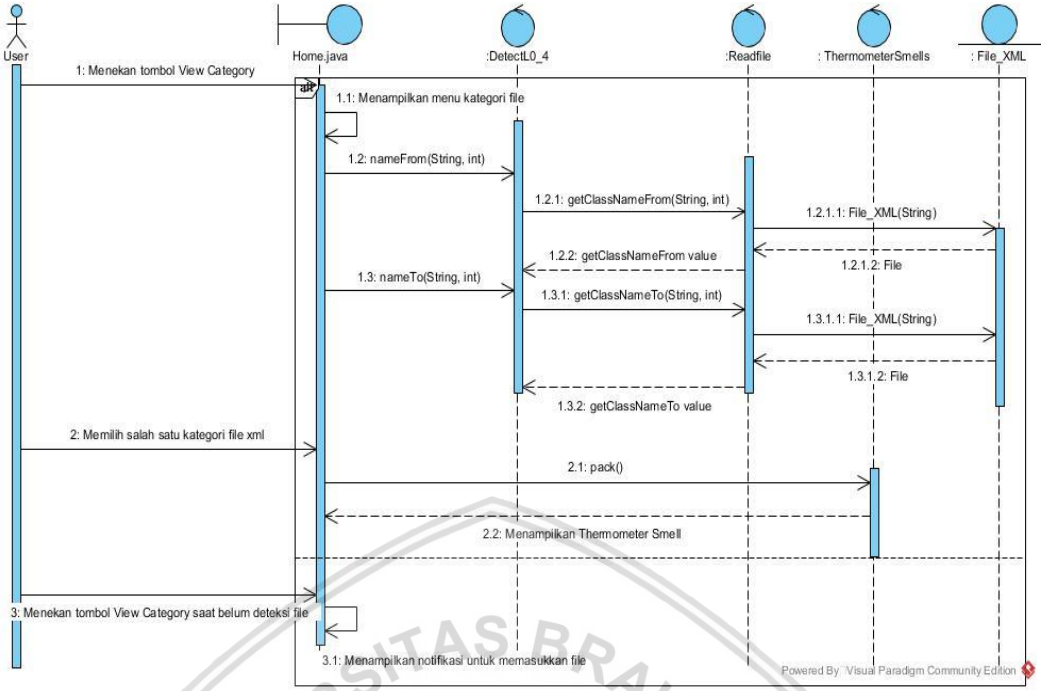
Pada Gambar 5.2 merupakan *sequence diagram* untuk mendeteksi file XML yang telah dimasukkan kedalam sistem. Langkah pertama yang dilakukan *user* adalah dengan menekan tombol *detect* pada tampilan sistem yang digambarkan dalam *sequence diagram* kedalam obyek *boundary* yaitu Home.java dan sistem akan melakukan deteksi file xml setelah sistem diberikan masukan. Pada proses tersebut, maka operasi yang ada pada obyek *controller* dijalankan, pada *sequence diagram* ini Readfile dan DetectL0_4 adalah obyek *controller*. Adapun obyek *entity* pada *sequence diagram* ini adalah File_XML.



Gambar 5.2 *Sequence Diagram* Mendeteksi File XML

5.1.1.1.3 Sequence Diagram Melihat Kategori File XML

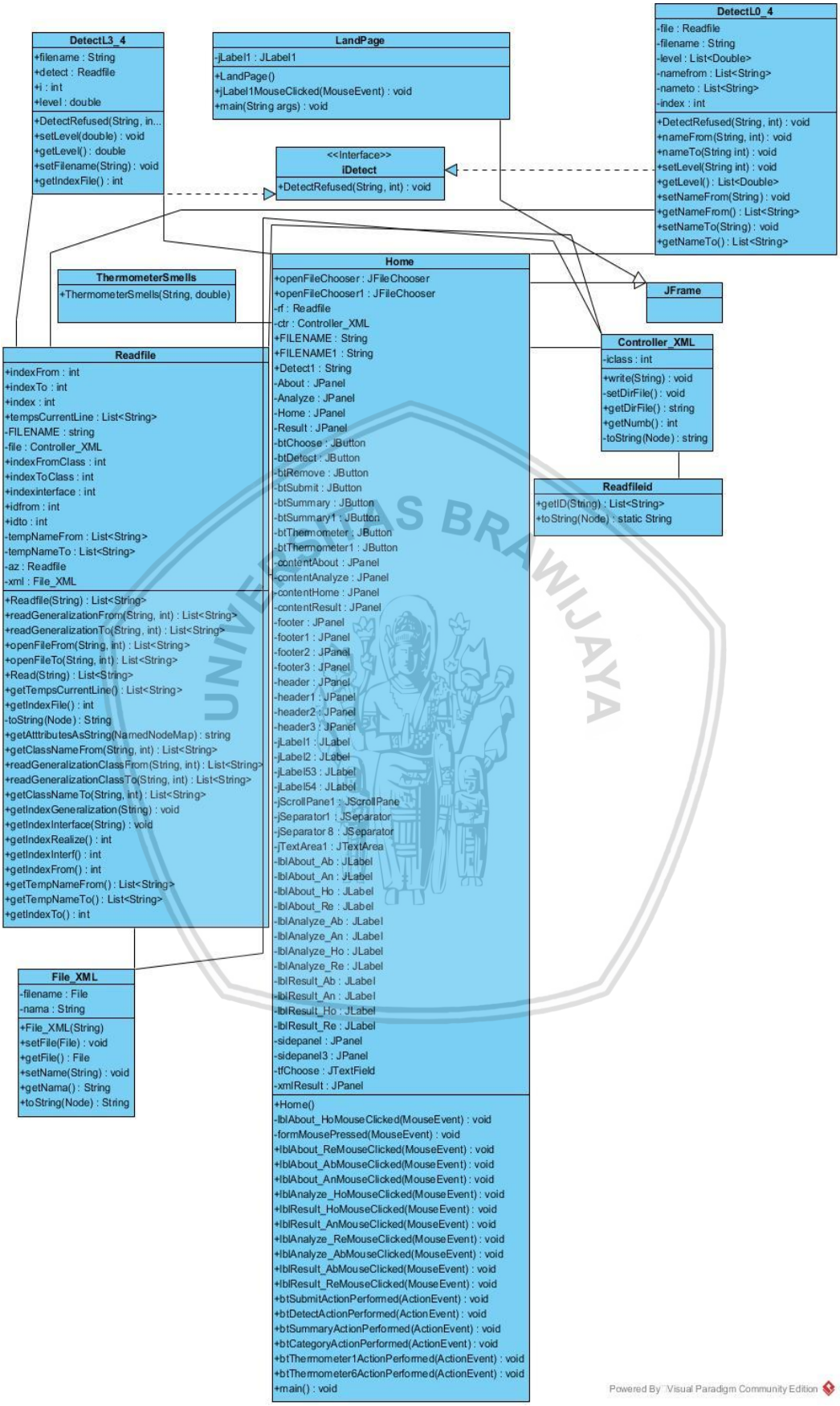
Pada Gambar 5.3 merupakan *sequence diagram* untuk melihat kategori file xml yang telah dilakukan deteksi *code smells refused bequest*. Langkah pertama yang dilakukan *user* adalah dengan menekan tombol *view category* pada tampilan sistem yang digambarkan dalam *sequence diagram* kedalam obyek *boundary* yaitu Home.java dan sistem akan menampilkan menu *view category*. Selanjutnya, ketika *user* memilih salah satu kategori file maka sistem akan menampilkan *thermometer smells* berdasarkan hasil deteksi *refused bequest* yang dilakukan, pada *sequence diagram* ini DetectL0_4, Readfile, ThermometerSmells adalah obyek *controller*. Adapun obyek *entity* pada *sequence diagram* ini adalah File_XML.



Gambar 5.3 *Sequence Diagram* Melihat Kategori File XML

5.1.1.2 Class Diagram

Pada Gambar 5.4 merupakan *class diagram* sistem Pendeteksian *Code Smells Refused Bequest*. *Class* diagram merupakan pemodelan statis untuk sebuah sistem, pada *class* diagram terdapat hubungan antar *class* dan turunannya (Rumbaugh, 2004). Pada setiap hubungan antar *class* memiliki jenis relasi tertentu berdasarkan kepada aturan penulisan *class* diagram. Pada Gambar 5.4 terdapat beberapa jenis *class* penyusun sistem antara lain adalah *class view* dan *class controller* sistem. Dimana *class view* merupakan *class* yang digunakan untuk berinteraksi dengan *user* antara lain adalah *class* LandPage.java dan Home.java yang merupakan *class* turunan dari JFrame. Adapun untuk *class controller* pada sistem sebagai penyusun alur logika sistem antara lain adalah DetectL3_4 dan DetectL0_4 yang mengimplementasikan fungsi pada *interface* iDetect. Adapun *class* lainnya adalah Thermometer Smells, Readfile, Controller_XML, Readfileid, Home, dan LandPage.



Gambar 5.4 *Class Diagram* Sistem Pendeteksian *Code Smells Refused Bequest*

5.1.2 Perancangan Komponen

Perancangan komponen merupakan penjabaran rincian sistem dari komponen perangkat lunak, menjelaskan rincian alur logika proses yang terjadi dalam komponen atau algoritma proses sistem. Pada perancangan komponen ini dipilih beberapa algoritma utama yang mewakili fungsi utama dalam sistem pendeteksian *code smells refused bequest.*

5.1.2.1 Perancangan Komponen *Method* Readfile

Nama *class* : Readfile.java – *Controller*
Nama operasi : Readfile(String)
Algoritma :

No	Pseudocode
1	Obyek fXmlFile diinisialisasi File dengan parameter input String
2	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
3	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag "Class" pada xml file nya
4	List<String> temp := new ArrayList<>() <- membuat array penampung hasil pencarian
5	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
6	inisialisasi setiap isi dokumen kedalam sebuah Node
7	if(Node := semua element dari xml)then
8	inisialisasi setiap isi Node kedalam obyek element
9	if(element tidak memiliki atribut "Background" & tidak memiliki atribut "Idref")then
10	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya.
11	end if
12	end if
13	end for
14	kembalikan nilai temp sebagai return pada operasi Readfile

5.1.2.2 Perancangan Komponen *Method* readGeneralizationFrom

Nama *class* : Readfile.java – *Controller*
Nama operasi : readGeneralizationFrom(String)
Algoritma :

No	Pseudocode
1	Obyek fXmlFile diinisialisasi File dengan parameter input String
2	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
3	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag "Generalization" pada xml file nya
4	List<String> temp := new ArrayList<>() <- membuat array penampung hasil pencarian
5	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
6	inisialisasi setiap isi dokumen kedalam sebuah Node
7	if(Node := semua element dari xml)then
8	inisialisasi setiap isi Node kedalam obyek element
9	if(element tidak memiliki atribut "Background" & tidak memiliki atribut "Idref" & memiliki atribut "Id")then
10	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya yang memiliki atribut "From".
11	end if
12	end if
13	end for
14	kembalikan nilai temp sebagai return pada operasi read GeneralizationFrom

5.1.2.3 Perancangan Komponen *Method* readGeneralizationTo

Nama *class* : Readfile.java – *Controller*
Nama operasi : readGeneralizationTo(String)
Algoritma :

No	Pseudocode
1	Obyek fXmlFile diinisialisasi File dengan parameter input String
2	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
3	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag "Generalization" pada xml file nya
4	List<String> temp := new ArrayList<>() <- membuat array penampung hasil pencarian
5	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
6	inisialisasi setiap isi dokumen kedalam sebuah Node
7	if(Node := semua element dari xml)then
8	inisialisasi setiap isi Node kedalam obyek element
9	if(element tidak memiliki atribut "Background" & tidak memiliki atribut "Idref" & memiliki atribut "Id")then
10	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya yang memiliki atribut "To".
11	end if
12	end if
13	end for
14	kembalikan nilai temp sebagai return pada operasi readGeneralizationTo

5.1.2.4 Perancangan Komponen *Method* openFileFrom

Nama *class* : Readfile.java – *Controller*
Nama operasi : openFileFrom(String)
Algoritma :

No	Pseudocode
1	Buat array penampung hasil keluaran sistem
2	for(setiap value pada method readGeneralizationFrom) then
3	inisialisasi obyek fXmlFile dengan path dokumen output dari masukan sistem setelah disubmit agar dapat membuka setiap class yang akan diidentifikasi
4	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
5	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag "Operation" pada xml file nya
6	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
7	inisialisasi setiap isi dokumen kedalam sebuah Node
8	if(Node := semua element dari xml)then
9	inisialisasi setiap isi Node kedalam obyek element
10	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya yang memiliki atribut "Name".
11	end if
12	end for
13	end for
14	kembalikan nilai filefrom sebagai return pada operasi openFileFrom

5.1.2.5 Perancangan Komponen *Method* openFileTo

Nama *class* : Readfile.java – *Controller*
Nama operasi : openFileTo(String)
Algoritma :

No	Pseudocode
1	Buat array penampung hasil keluaran sistem
2	for(setiap value pada method readGeneralizationTo) then
3	inisialisasi obyek fXmlFile dengan path dokumen output dari masukan sistem setelah disubmit agar dapat membuka setiap class yang akan diidentifikasi
4	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
5	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag "Operation" pada xml file nya
6	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
7	inisialisasi setiap isi dokumen kedalam sebuah Node
8	if(Node := semua element dari xml)then
9	inisialisasi setiap isi Node kedalam obyek element
10	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya yang memiliki atribut "Name".
11	end if
12	end for
13	end for
14	kembalikan nilai filefrom sebagai return pada operasi openFileTo

5.1.2.6 Perancangan Komponen *Method* DetectRefused

Nama *class* : DetectL0_4.java – *Controller*

Nama operasi : DetectRefused(String, int)

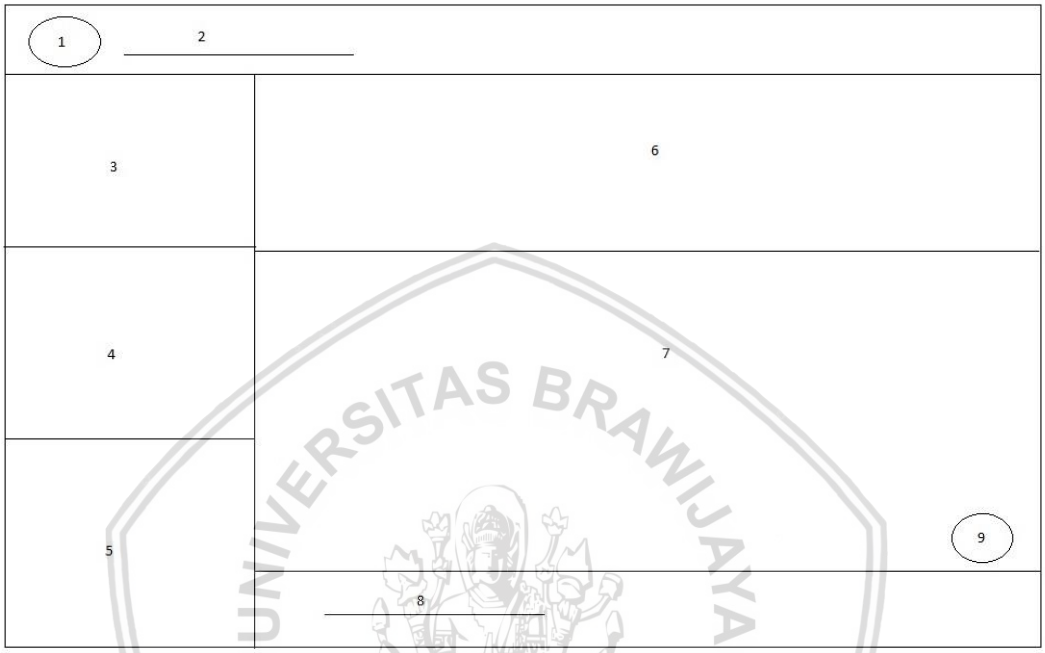
Algoritma :

No	Pseudocode
1	Buat variabel untuk menampung nilai deteksi yang sama
2	Buat array untuk menampung nilai yang sama dari variabel sebelumnya
3	for (int h = 1 dimana h hingga jumlah generalization dan h bertambah 1 setiap loop) then
4	Masukkan nilai dari pemanggilan openFileTo kedalam sebuah array
5	Masukkan nilai dari pemanggilan openFileFrom kedalam sebuah array
6	for (int i = 0 dimana i hingga jumlah size array from dan i bertambah 1 setiap loop) then
7	for (int j = 0 dimana i hingga jumlah size array to dan j bertambah 1 setiap loop) then
8	if(nilai from sama dengan nilai to) then
9	hitung jumlah yang sama
10	end if
11	end for
12	inisialisasi array penampung nilai sama berdasarkan jumlah hasil deteksi
13	nilai penghitung diset 0 setiap loop dilakukan
14	end for
15	buat variabel untuk mengecek nilai dari setiap hasil deteksi yang sama
16	buat variabel untuk menghitung totalnya
17	for(untuk setiap nilai yang sama) then
18	nilai total diinisialisasi nilai yang sama tersebut
19	melakukan pengecekan index dari loop
20	end for
21	if(total = 0) then
22	setLevel(3.0)
23	else if (aa >= 1 && aa >= (indexcheck / 2) && aa < indexcheck)then
24	setLevel(1.0)
25	else if (aa >= 1 && aa <= (indexcheck / 2))then
26	setLevel(2.0)
27	else if (aa == indexcheck) then
28	setLevel(0.0)
29	end if
30	clear setiap nilai penghitung
31	end for

5.1.3 Perancangan Antarmuka

Pada bagian ini akan dijelaskan perancangan antarmuka yang akan digunakan sebagai acuan untuk implementasi sistem berupa visualisasi antarmuka kedalam bentuk rancangan dasar beserta detail penjelasan dari setiap komponen antarmuka.

1. Perancangan Antarmuka Tampilan Utama Halaman Home



Gambar 5.5 Perancangan Antarmuka Tampilan Utama Halaman Home

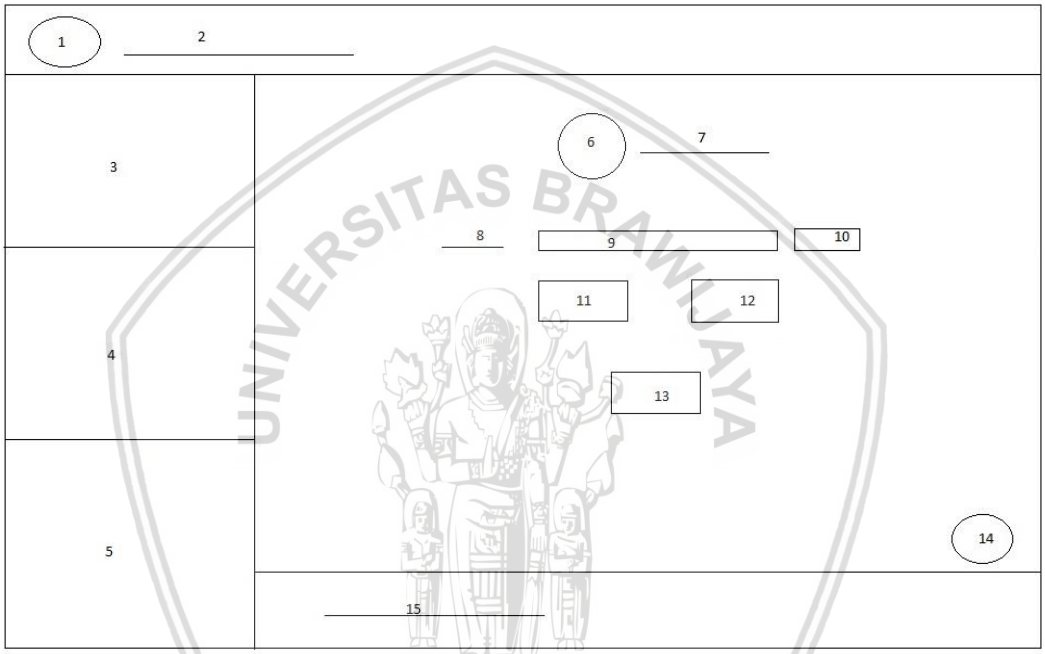
Pada Gambar 5.5 merupakan perancangan antarmuka tampilan utama halaman home dan diberikan penjelasan terhadap rancangan tersebut pada Tabel 5.1 berikut.

Tabel 5.1 Detail Perancangan Antarmuka Tampilan Utama Halaman Home

No	Nama Objek	Tipe	Deskripsi
1	Logo Sistem	jLabel	Logo Sistem.
2	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
3	Menu *Analyze*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Analyze* untuk mendeteksi *Refused Bequest*.
4	Menu *Result*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Result* untuk melihat hasil deteksi *Refused Bequest*.
5	Menu *About*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *About*.
6	Content Home	jPanel	Untuk menampilkan content dari tampilan utama halaman Home.

7	Content Home	jPanel	Untuk menampilkan content dari tampilan utama halaman Home.
8	Content Footer	jLabel	Untuk menampilkan informasi footer.
9	Logo Sistem	jLabel	Untuk menampilkan logo sistem pada panel content Home.

2. Perancangan Antarmuka Menu Analyze



Gambar 5.6 Perancangan Antarmuka Menu Analyze

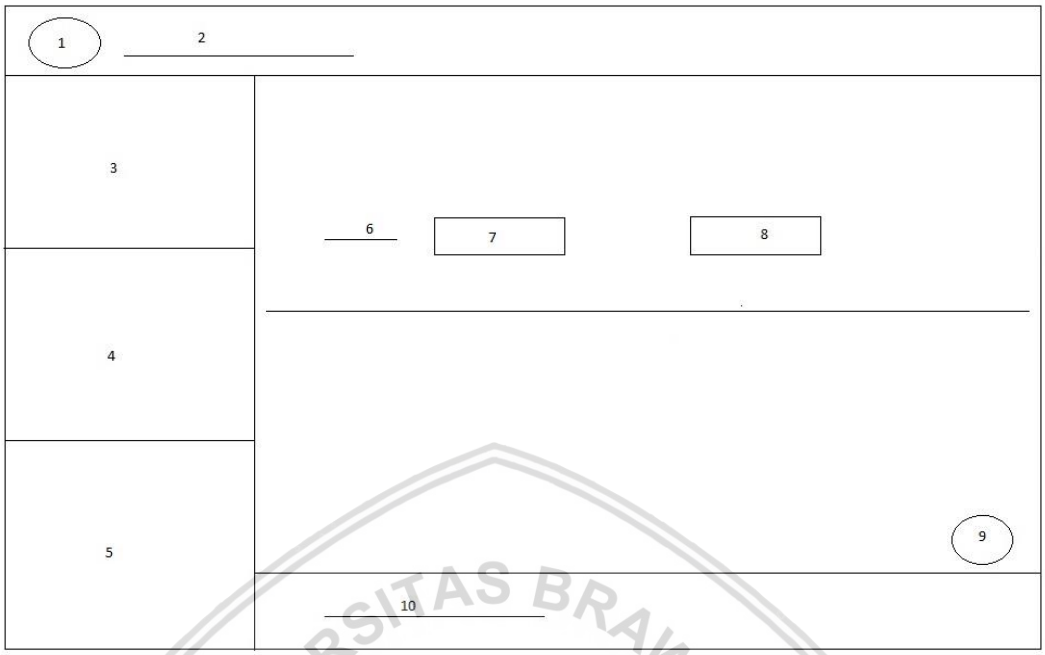
Pada Gambar 5.6 merupakan perancangan antarmuka menu *analyze* dan diberikan penjelasan terhadap rancangan tersebut pada Tabel 5.2 berikut.

Tabel 5.2 Detail Perancangan Antarmuka Menu Analyze

No	Nama Objek	Tipe	Deskripsi
1	Logo Sistem	jLabel	Logo Sistem.
2	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
3	Menu *Analyze*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Analyze* untuk mendeteksi *Refused Bequest*.
4	Menu *Result*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Result* untuk melihat hasil deteksi *Refused Bequest*.

5	Menu *About*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *About*.
6	Logo Utama	jLabel	Logo Utama Sistem.
7	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
8	Keterangan Jenis File	jLabel	Untuk memberikan keterangan masukan sistem berupa file XML.
9	tfChooser	Text Field	Untuk menampilkan nama file yang dipilih untuk dimasukan kedalam sistem atau informasi terkait masukan sistem *Refused Bequest*.
10	Choose File	Button	Tombol yang digunakan untuk membuka *open dialog* file yang akan dimasukan kedalam sistem.
11	Submit File	Button	Tombol yang digunakan untuk masukan sistem setelah file dipilih.
12	Remove File	Button	Tombol yang digunakan untuk menghapus file yang dipilih.
13	Detect File	Button	Tombol yang digunakan untuk mendeteksi file yang dimasukkan kedalam sistem.
14	Logo Sistem	jLabel	Untuk menampilkan logo sistem pada content *Analyze*.
15	Content Footer	jLabel	Untuk menampilkan informasi footer.

3. Perancangan Antarmuka Menu Result



Gambar 5.7 Perancangan Antarmuka Menu Result

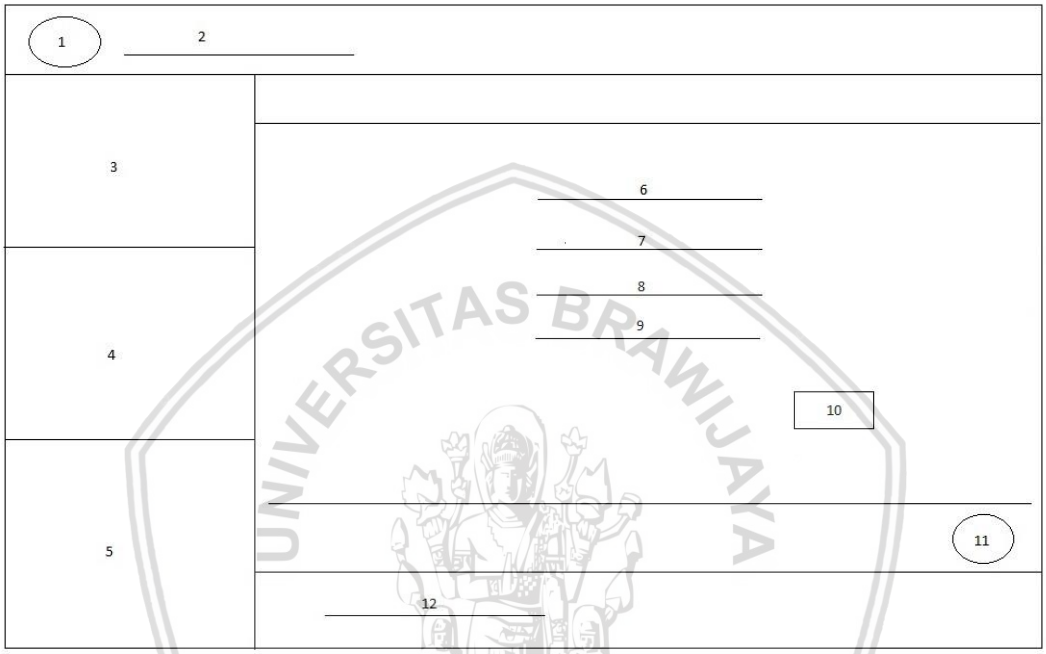
Pada Gambar 5.7 merupakan perancangan antarmuka menu *result* dan diberikan penjelasan terhadap rancangan tersebut pada tabel 5.3 berikut.

Tabel 5.3 Detail Perancangan Antarmuka Menu Result

No	Nama Objek	Tipe	Deskripsi
1	Logo Sistem	jLabel	Logo Sistem.
2	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
3	Menu *Analyze*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Analyze* untuk mendeteksi *Refused Bequest*.
4	Menu *Result*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Result* untuk melihat hasil deteksi *Refused Bequest*.
5	Menu *About*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *About*.
6	Keterangan Jenis File	jLabel	Untuk memberikan keterangan masukan sistem berupa file XML.
7	Summary	Button	Tombol untuk menampilkan *summary* hasil deteksi *refused bequest*.
8	Kategori	Button	Tombol untuk menampilkan kategori berdasarkan *termometer smells* hasil

			deteksi *refused bequest*.
9	Logo Sistem	jLabel	Untuk menampilkan logo sistem pada panel content Home.
10	Content Footer	jLabel	Untuk menampilkan informasi footer.

4. Perancangan Antarmuka Menu View Summary



Gambar 5.8 Perancangan Antarmuka Menu View Summary

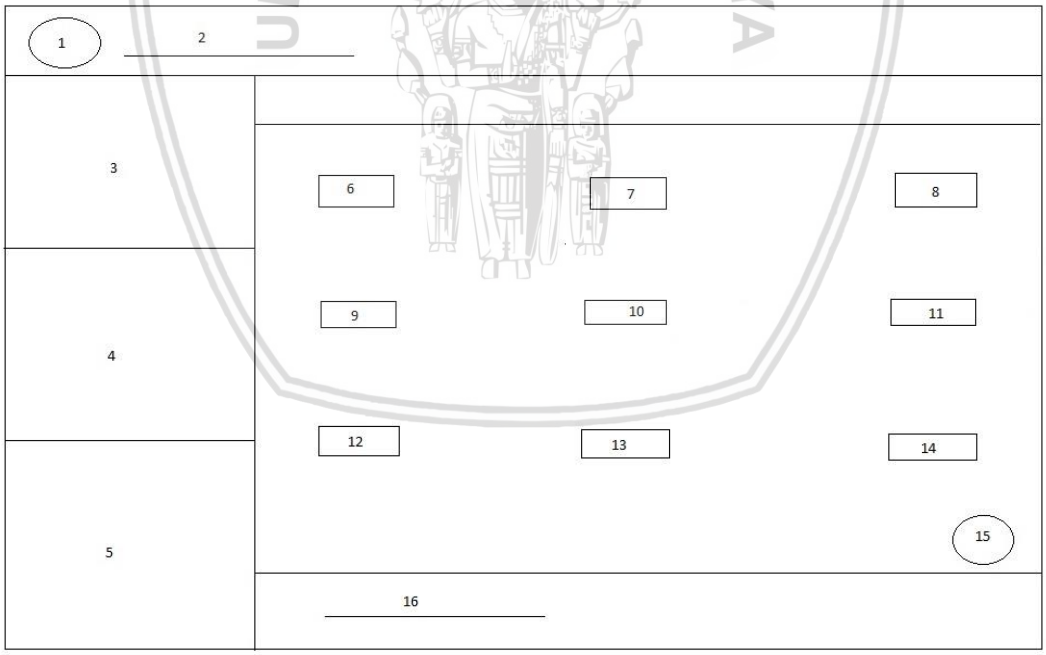
Pada Gambar 5.8 merupakan perancangan antarmuka menu *view summary* dan diberikan penjelasan terhadap rancangan tersebut pada Tabel 5.4 berikut.

Tabel 5.4 Detail Perancangan Antarmuka Menu Result

No	Nama Objek	Tipe	Deskripsi
1	Logo Sistem	jLabel	Logo Sistem.
2	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
3	Menu *Analyze*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Analyze* untuk mendeteksi *Refused Bequest*.
4	Menu *Result*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Result* untuk melihat hasil deteksi *Refused Bequest*.
5	Menu *About*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *About*.
6	Summary1	jLabel	Untuk menampilkan hasil

			deteksi nama *class.*
7	Summary2	jLabel	Untuk menampilkan hasil deteksi jumlah *class* yang dideteksi.
8	Summary3	jLabel	Untuk menampilkan hasil deteksi jumlah *Generalization* yang dideteksi.
9	Summary4	jLabel	Untuk menampilkan hasil deteksi jumlah *interface* yang dideteksi.
10	Back	Button	Tombol untuk melakukan kembali kehalaman *Result*.
11	Logo Sistem	jLabel	Untuk menampilkan logo sistem pada panel content Home.
12	Content Footer	jLabel	Untuk menampilkan informasi footer.

5. Perancangan Antarmuka Menu View Category



Gambar 5.9 Perancangan Antarmuka Menu View Category

Pada Gambar 5.9 merupakan perancangan antarmuka menu *view category* dan diberikan penjelasan terhadap rancangan tersebut pada Tabel 5.5 berikut.

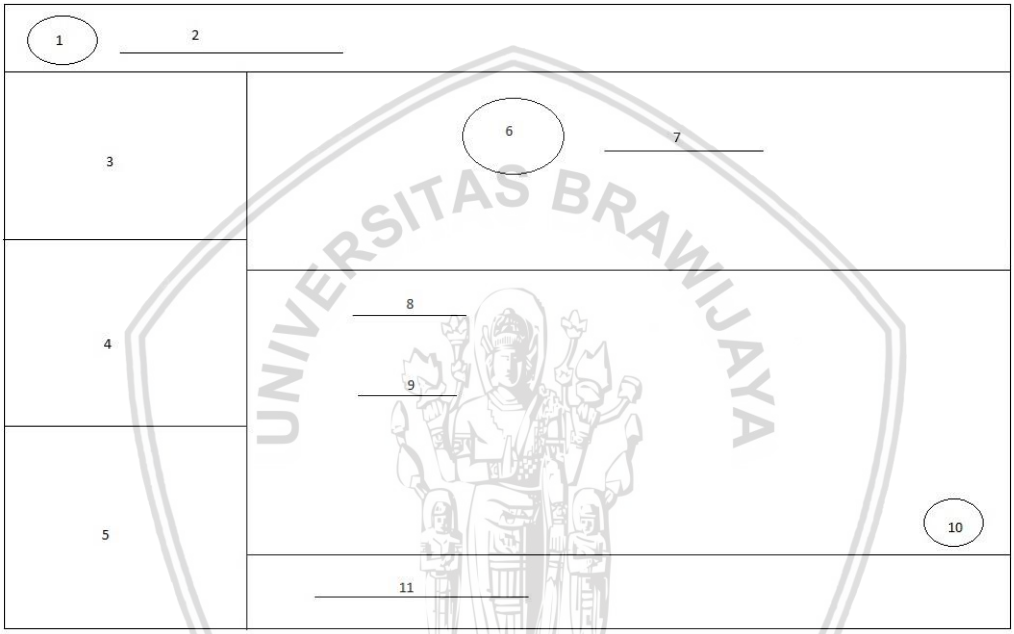
Tabel 5.5 Detail Perancangan Antarmuka Menu Result

No	Nama Objek	Tipe	Deskripsi
1	Logo Sistem	jLabel	Logo Sistem.
2	Judul Sistem	jLabel	Judul Sistem *Refused*

			Bequest.
3	Menu *Analyze*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Analyze* untuk mendeteksi *Refused Bequest*.
4	Menu *Result*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Result* untuk melihat hasil deteksi *Refused Bequest*.
5	Menu *About*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *About*.
6	Kategori1	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
7	Kategori2	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
8	Kategori3	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
9	Kategori4	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
10	Kategori5	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
11	Kategori6	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
12	Kategori7	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
13	Kategori8	Button	Tombol untuk menampilkan Kategori kedalam Termometer Smell.
14	Kategori9	Button	Tombol untuk

			menampilkan Kategori kedalam Termometer Smell.
15	Logo Sistem	jLabel	Untuk menampilkan logo sistem pada panel content Home dan kembali kehalaman Result.
16	Content Footer	jLabel	Untuk menampilkan informasi footer.

6. Perancangan Antarmuka Menu About



Gambar 5.10 Perancangan Antarmuka Menu About

Pada Gambar 5.10 merupakan perancangan antarmuka menu *about* dan diberikan penjelasan terhadap rancangan tersebut pada Tabel 5.6 berikut.

Tabel 5.6 Detail Perancangan Antarmuka Menu About

No	Nama Objek	Tipe	Deskripsi
1	Logo Sistem	jLabel	Logo Sistem.
2	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
3	Menu *Analyze*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Analyze* untuk mendeteksi *Refused Bequest*.
4	Menu *Result*	jPanel, jLabel	Menu yang digunakan untuk mengakses Menu *Result* untuk melihat hasil deteksi *Refused Bequest*.
5	Menu *About*	jPanel, jLabel	Menu yang digunakan

			untuk mengakses Menu *About*.
6	Logo Utama	jLabel	Logo Utama Sistem.
7	Judul Sistem	jLabel	Judul Sistem *Refused Bequest.*
8	Content About	jLabel	Untuk menampilkan content dari menu *about*.
9	Content About	jLabel	Untuk menampilkan content dari menu *about*.
10	Logo Sistem	jLabel	Untuk menampilkan logo sistem pada panel content Home.
11	Content Footer	jLabel	Untuk menampilkan informasi footer.

5.2 Implementasi Sistem

Proses implementasi merupakan proses tahapan selanjutnya setelah perancangan sistem telah selesai dilakukan. Proses implementasi merupakan tahapan dalam mengembangkan perangkat lunak. Setiap kebutuhan yang telah digali dan didapatkan maka selanjutnya akan diimplementasikan pada pembangunan sistem. Implementasi sistem pada penelitian ini meliputi struktur dan kode program berdasarkan algoritma pada bagian perancangan komponen. Pada bagian ini akan diuraikan mengenai spesifikasi sistem, implementasi kode program, serta implementasi antarmuka sistem.

5.2.1 Spesifikasi Sistem

Pada bagian ini akan diuraikan mengenai spesifikasi sistem yang telah digunakan untuk membangun sistem pendeteksian *refused bequest*. Spesifikasi yang akan diuraikan dimuat kedalam beberapa poin, antara lain adalah spesifikasi perangkat keras, perangkat lunak, serta sistem operasi.

Spesifikasi perangakat keras yang digunakan terdapat dalam Tabel 5.7 berikut.

Tabel 5.7 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Laptop	1. Intel® Core™ i7-6700HQ CPU @2.60Ghz (8 CPUs) 2. *Memory*(RAM) 8GB 3. 1 TB HDD 7200 RPM

Spesifikasi perangakat lunak yang digunakan terdapat dalam Tabel 5.8 berikut.

Tabel 5.8 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Editor Dokumen	Microsoft Word 2016
Editor Perancangan	Visual Paradigm *Community*

	Edition
Editor Pemrograman	Netbeans IDE 8.0.2
Bahasa Pemrograman	Java™
Libraries	Jfreechart-1.0.19

Spesifikasi sistem operasi yang digunakan terdapat dalam Tabel 5.9 berikut.

Tabel 5.9 Sistem Operasi

Nama Komponen	Spesifikasi
Sistem Operasi Pengembangan Sistem	Windows 10 Pro x64
Sistem Operasi Pengujian Sistem	Windows 10 Pro x64

5.2.2 Implementasi Kode Program

Pada bagian ini merupakan implementasi kode program dimana pada bagian perancangan komponen yang telah dibuat berupa algorita dalam bentu *pseudocode* diubah kedalam bahasa pemrograman. Adapun pada implementasi kode program ini merubah *pseudocode* kedalam bahasa pemrograman Java sebagai bahasa yang digunakan dalam membangun sistem pendeteksian *code smells refused bequest.*

5.2.2.1 Implementasi Kode Program *Method* Readfile

Nama *class* : Readfile.java – *Controller*
Nama operasi : Readfile(String)
Algoritma :

Tabel 5.10 Implementasi Kode Program *Method* Readfile

No	Source code
1	`File fXmlFile = new File(namefile);`
2	`DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();`
3	`DocumentBuilder db = dbf.newDocumentBuilder();`
4	`Document doc = db.parse(fXmlFile);`
5	`doc.getDocumentElement();`
6	`NodeList nList = doc.getElementsByTagName("Class");`
7	`List<String> temp = new ArrayList<>();`
8	`for (int i = 0; i < nList.getLength(); i++) {`
9	`Node nNode = nList.item(i);`
10	`if (nNode.getNodeType() == Node.ELEMENT_NODE) {`
11	`Element eElement = (Element) nNode;`
12	`if (!eElement.hasAttribute("Background") && !eElement.hasAttribute("Idref")) {`
13	`temp.add(toString(eElement));`
14	`}`
15	`}`
16	`}`
17	`//  for (String a : temp) {`
18	`//  System.out.println(a);`
19	`//  }`
20	`return temp;`

5.2.2.2 Implementasi Kode Program *Method* readGeneralizationFrom

Nama *class* : Readfile.java – *Controller*
Nama operasi : readGeneralizationFrom(String)
Algoritma :

Tabel 5.11 Implementasi Kode Program *Method* readGeneralizationFrom

No	Source code
1	`File fXmlFile = new File(namefile);`
2	`DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();`
3	`DocumentBuilder db = dbf.newDocumentBuilder();`
4	`Document doc = db.parse(fXmlFile);`
5	`doc.getDocumentElement();`
6	`NodeList nList = doc.getElementsByTagName("Generalization");`
7	`List<String> tempFrom = new ArrayList<>();`
8	`for (int i = 0; i < nList.getLength(); i++) {`
9	`Node nNode = nList.item(i);`
10	
11	`if (nNode.getNodeType() == Node.ELEMENT_NODE) {`
12	`Element eElement = (Element) nNode;`
13	`if (!eElement.hasAttribute("Background") && !eElement.hasAttribute("Idref") && eElement.hasAttribute("Id")) {`
14	`//indexFrom++;`
15	`//System.out.println(indexFrom);`
16	`tempFrom.add(eElement.getAttribute("From"));`
	`//tempFrom.add(eElement.getAttribute("To"));`
	`//`
17	`//System.out.println(eElement.getAttribute("From"));`
18	`//`
19	`System.out.println(eElement.getAttribute("To"));`
20	`}`
21	`}`
22	`}`
23	`//for (String a : tempFrom) {`
24	`//System.out.println(a);`
25	`//}`
	`return tempFrom;`

5.2.2.3 Implementasi Kode Program *Method* readGeneralizationTo

Nama *class* : Readfile.java – *Controller*
Nama operasi : readGeneralizationTo(String)
Algoritma :

Tabel 5.12 Implementasi Kode Program *Method* readGeneralizationTo

No	Source code
1	`File fXmlFile = new File(namefile);`
2	`DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();`
3	`DocumentBuilder db = dbf.newDocumentBuilder();`
4	`Document doc = db.parse(fXmlFile);`
5	`doc.getDocumentElement();`
6	`NodeList nList = doc.getElementsByTagName("Generalization");`
7	`List<String> tempto = new ArrayList<>();`
8	`for (int i = 0; i < nList.getLength(); i++) {`
9	`Node nNode = nList.item(i);`
10	`if (nNode.getNodeType() == Node.ELEMENT_NODE) {`

```
11                  Element eElement = (Element) nNode;
12                  if (!eElement.hasAttribute("Background") &&
    !eElement.hasAttribute("Idref") && eElement.hasAttribute("Id")) {
13                      tempto.add(eElement.getAttribute("To"));
14                  }
15              }
16          }
17  //        for (String a : temp) {
18  //            System.out.println(a);
19  //        }
20          return tempto;
```

5.2.2.4 Implementasi Kode Program *Method* openFileFrom

Nama *class* : Readfile.java – *Controller*
Nama operasi : openFileFrom(String)
Algoritma :

Tabel 5.13 Implementasi Kode Program *Method* openFileFrom

No	Source code
1	`az = new Readfile();`
2	`List<String> filefrom = new ArrayList<>();`
3	`for (String a : az.readGeneralizationFrom(filename, idfrom))`
4	`{`
5	`File fXmlFile = new File(file.getDirFile() + "\\output\\" + a + ".xml");`
6	`DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();`
7	`DocumentBuilder db = dbf.newDocumentBuilder();`
8	`Document doc = db.parse(fXmlFile);`
9	`doc.getDocumentElement();`
10	`NodeList nList = doc.getElementsByTagName("Operation");`
11	`for (int i = 0; i < nList.getLength(); i++) {`
12	`Node nNode = nList.item(i);`
13	`if (nNode.getNodeType() == Node.ELEMENT_NODE) {`
14	`Element eElement = (Element) nNode;`
15	`filefrom.add(eElement.getAttribute("Name"));`
16	`}`
17	`}`
18	`}`
	`return filefrom;`

5.2.2.5 Implementasi Kode Program *Method* openFileTo

Nama *class* : Readfile.java – *Controller*
Nama operasi : openFileTo(String)
Algoritma :

Tabel 5.14 Implementasi Kode Program *Method* openFileTo

No	Source code
1	`Readfile az = new Readfile();`
2	`List<String> fileto = new ArrayList<>();`
3	`for (String a : az.readGeneralizationTo(filename, idto)) {`
4	`File fXmlFile = new File(file.getDirFile() + "\\output\\" + a + ".xml");`
5	`DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();`
6	`DocumentBuilder db = dbf.newDocumentBuilder();`
7	`Document doc = db.parse(fXmlFile);`
8	`doc.getDocumentElement();`
9	`NodeList nList = doc.getElementsByTagName("Operation");`
10	`for (int i = 0; i < nList.getLength(); i++) {`
11	`Node nNode = nList.item(i);`

```
12                  if (nNode.getNodeType() == Node.ELEMENT_NODE) {
13                      Element eElement = (Element) nNode;
14                      fileto.add(eElement.getAttribute("Name"));
15                  }
16              }
17          }
18          return fileto;
```

5.2.2.6 Implementasi Kode Program *Method* DetectRefused

Nama *class* : DetectL0_4.java – *Controller*
Nama operasi : DetectRefused(String, int)
Algoritma :

Tabel 5.15 Implementasi Kode Program *Method* DetectRefused

No	Pseudocode
1	`try {`
2	`int x = 0;`
3	`List<Integer> tempyes = new ArrayList<>();`
4	`for (int h = 1; h <= indexRealize; h++) {`
5	`List<String> to = file.openFileTo(filename, h);`
6	`List<String> from = file.openFileFrom(filename, h);`
7	`for (int i = 0; i < from.size(); i++) {`
8	`for (int j = 0; j < to.size(); j++) {`
9	`if (from.get(i).equals(to.get(j))) {`
10	`x++;`
11	`} else {`
12	`}`
13	`}`
14	`tempyes.add(x);`
15	`x = 0;`
16	`}`
17	`int indexcheck = 0;`
18	`int aa = 0;`
19	`for (int a : tempyes) {`
20	`aa += a;`
21	`indexcheck++;`
22	`}`
23	`if (aa == 0) {`
24	`setLevel(3.0);`
25	`} else if (aa >= 1 && aa >= (indexcheck / 2) && aa < indexcheck) {`
26	`setLevel(1.0);`
27	`} else if (aa >= 1 && aa <= (indexcheck / 2)) {`
28	`setLevel(2.0);`
29	`} else if (aa == indexcheck) {`
30	`setLevel(0.0);`
31	`}`
32	`aa = 0;`
33	`from.clear();`
34	`to.clear();`
35	`tempyes.clear();`
36	`}`
37	`} catch (ParserConfigurationException | SAXException |`
38	`IOException | TransformerException ex) {`
39	`ex.printStackTrace();`
40	`}`

5.2.3 Implementasi Antarmuka

Pada bagian ini akan ditampilkan antarmuka hasil implementasi antarmuka sistem yang berdasar kepada rancangan sistem yang telah dibuat sebelumnya pada bagian perancangan antarmuka.

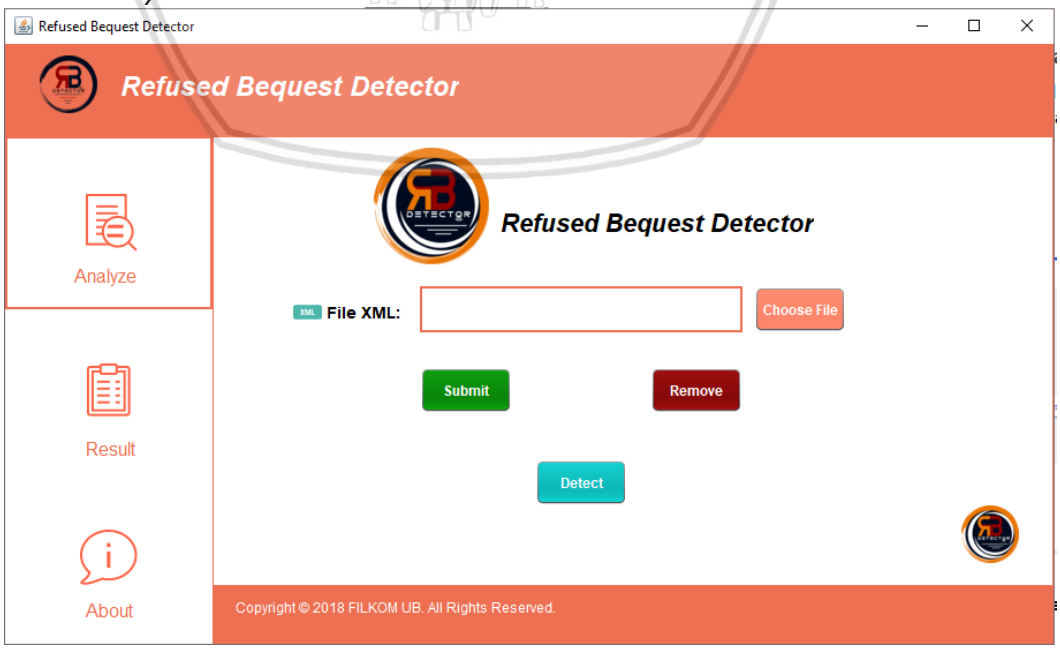
1. Antarmuka Tampilan Utama Halaman Home

   Pada Gambar 5.11 berikut akan ditampilkan hasil implementasi antarmuka tampilan utama halaman *home*.

Gambar 5.11 Implementasi Antarmuka Tampilan Utama Halaman Home

2. Antarmuka Menu *Analyze*

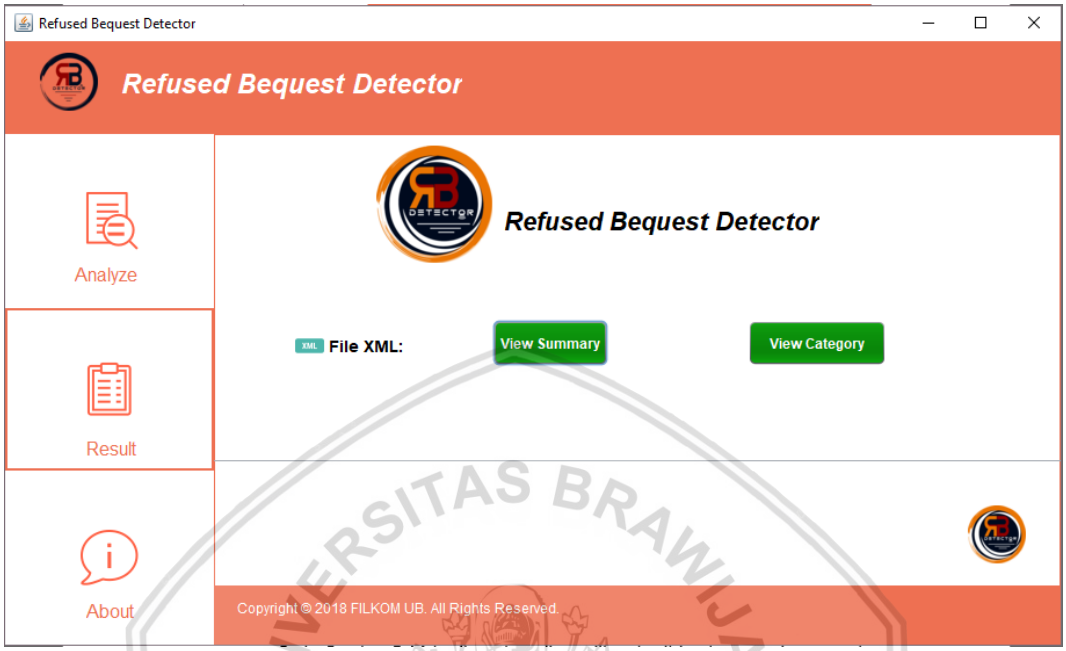
   Pada Gambar 5.12 berikut akan ditampilkan hasil implementasi antarmuka menu *analyze*.



Gambar 5.12 Implementasi Antarmuka Menu *Analyze*

3. Antarmuka Menu *Result*

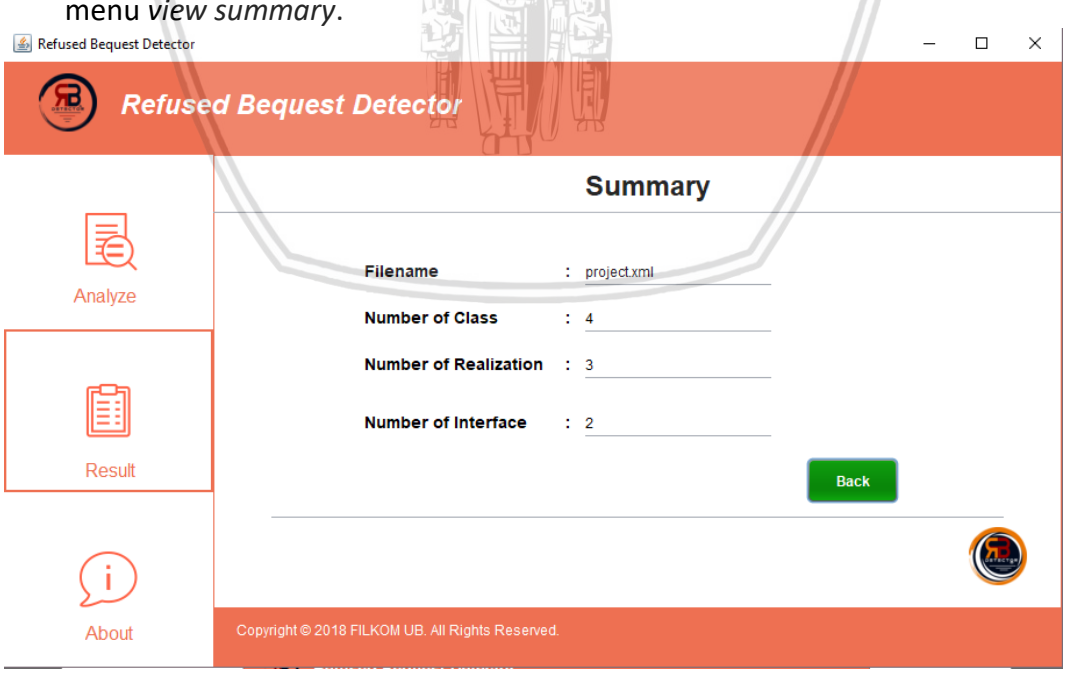
   Pada Gambar 5.13 berikut akan ditampilkan hasil implementasi antarmuka menu *result*.



Gambar 5.13 Implementasi Antarmuka Menu *Result*

4. Antarmuka Menu *View Summary*

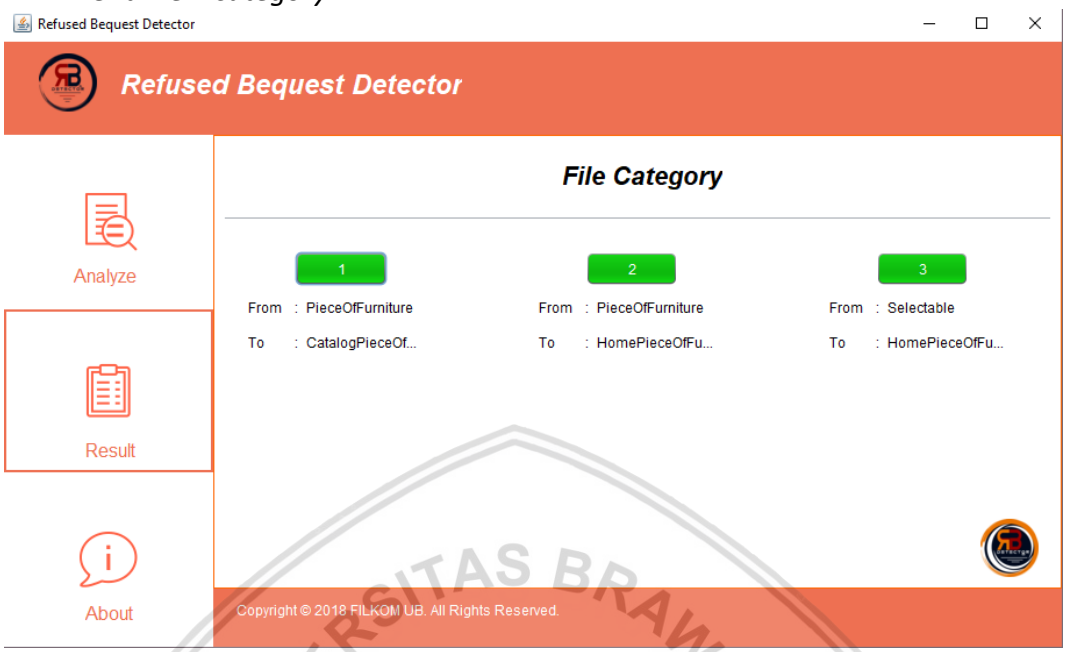
   Pada Gambar 5.14 berikut akan ditampilkan hasil implementasi antarmuka menu *view summary*.



Gambar 5.14 Implementasi Antarmuka Menu *View Summary*

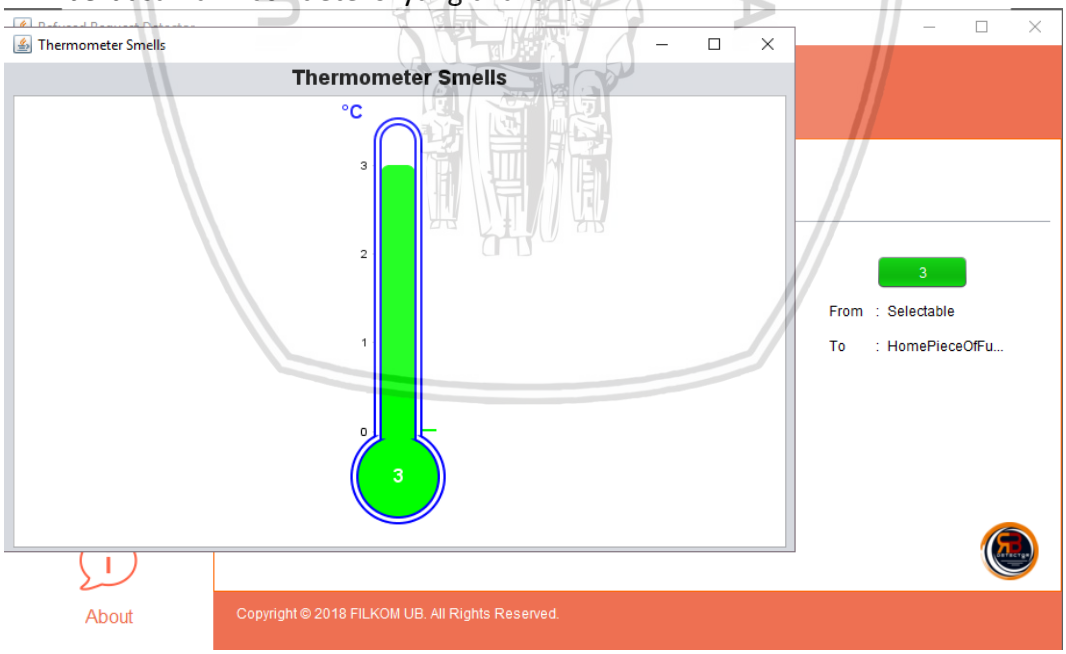
5. Antarmuka Menu *View Category*

Pada Gambar 5.15 berikut akan ditampilkan hasil implementasi antarmuka menu *view category*.



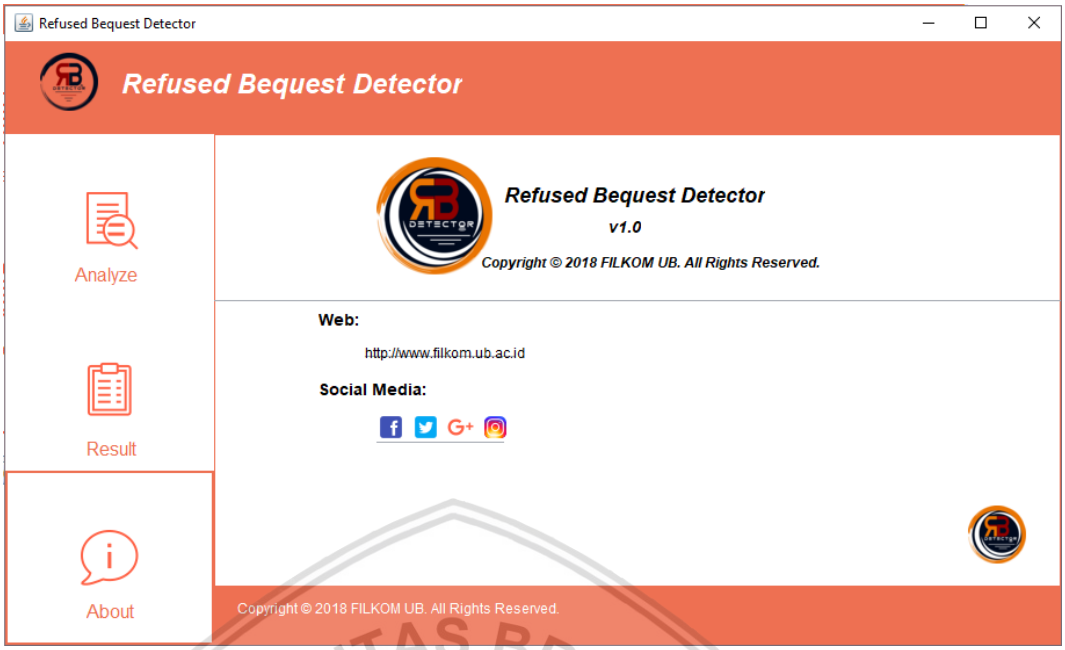
Gambar 5.15 Implementasi Antarmuka Menu *View Category*

6. Antarmuka *Thermometer Smells*
   Pada Gambar 5.16 berikut akan ditampilkan antarmuka *thermometer smells* berdasarkan hasil deteksi yang dilakukan.



Gambar 5.16 Implementasi Antarmuka *Thermometer Smells*

7. Antarmuka Menu *About*
   Pada gambar 5.17 berikut akan ditampilkan hasil implementasi antarmuka menu *about*.



Gambar 5.17 Implementasi Antarmuka Menu *About*

BAB 6 PENGUJIAN SISTEM

Pada tahap pengujian sistem dilakukan setelah melakukan implementasi sistem. Pengujian bertujuan untuk melakukan pemeriksaan terhadap implementasi yang dilakukan apakah telah sesuai dengan analisis kebutuhan dan perancangan sistem. Tahap pengujian yang akan dilakukan pada penelitian ini adalah pengujian unit, pengujian integrase dan pengujian validasi sistem.

6.1 Pengujian Unit

Pengujian unit merupakan pengujian yang berfokus kepada unit (komponen) dari perangkat lunak yang akan diuji berdasarkan implementasi yang telah dilakukan. Pada pengujian unit yang dilakukan ini digunakan metode *whitebox testing* dengan teknik pengujian *basis path testing*. Metode ini bertujuan untuk memastikan bahwa setidaknya setiap jalur alur logika sistem dapat dijalankan sebanyak satu kali dalam menjalankannya. Adapun langkah-langkah yang digunakan dalam membuat kasus uji pada teknik *basis path testing* ini adalah sebagai berikut:

1. Membuat diagram alir berdasarkan algoritma sistem yang akan diuji.
2. Menentukan nilai *cyclomatic complexity* dari diagram alir yang telah dibuat.
3. Menentukan *independent path* berdasarkan diagram alir yang telah dibuat yang akan digunakan sebagai kasus uji sistem.

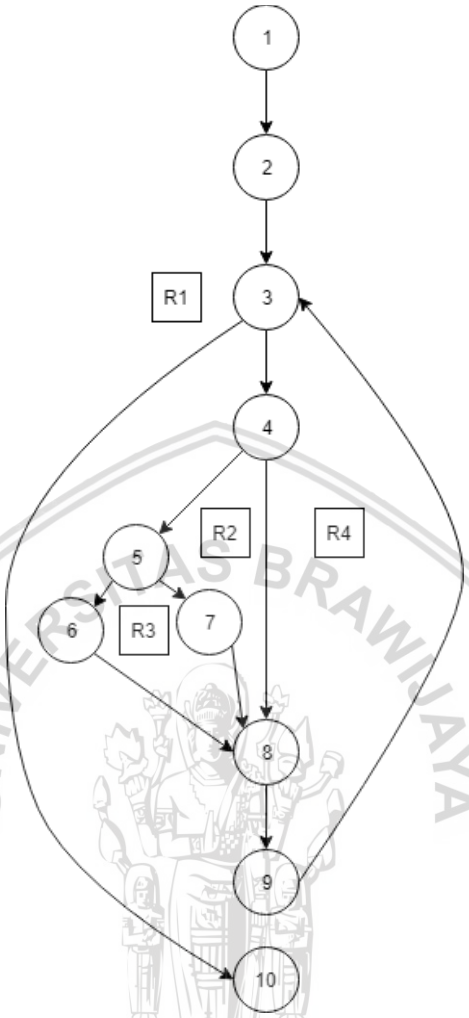
6.1.2 Pengujian Unit *Method* Readfile

1. *Pseudocode*

No	Pseudocode
1	Obyek fXmlFile diinisialisasi File dengan parameter input String
1	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
2	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag "Class" pada xml file nya
2	List<String> temp := new ArrayList<>() <- membuat array penampung hasil pencarian
3	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
4	inisialisasi setiap isi dokumen kedalam sebuah Node
4	if(Node := semua element dari xml)then
5	inisialisasi setiap isi Node kedalam obyek element
5	if(element tidak memiliki atribut "Background" & tidak memiliki atribut "Idref")then
6	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya.
7	end if
8	end if
9	end for
10	kembalikan nilai temp sebagai return pada operasi Readfile

2. *Basis Path Testing*

2.1 Flow Graph



Gambar 6.1 *Flow Graph Method* Readfile

2.2 Cyclomatic Complexity (V(G))

- V(G) = Jumlah Region = 4
- V(G) = Jumlah edge – jumlah node + 2 = (12 – 10) + 2 = 4
- V(G) = Jumlah predicate node + 1 = 3 + 1 = 4

2.3 Independent Path

- Jalur 1 : 1-2-3-10
- Jalur 2 : 1-2-3-4-8-9-3-10
- Jalur 3 : 1-2-3-4-5-7-8-9-3-10
- Jalur 4 : 1-2-3-4-5-6-8-9-3-10

Tabel 6.1 Hasil pengujian unit *method* Readfile

No	No. Jalur	Data Pengujian	Expected Result	Result	Status
1	1	Jika file yang dideteksi tidak mengandung tag "Class".	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*
2	2	Jika file yang dideteksi mengandung tag "Class" namun kondisi if bernilai *false.*	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*
3	3	Jika file yang dideteksi mengandung tag "Class" kondisi if bernilai true sedangkan kondisi *nested if* bernilai *false.*	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*
4	4	Jika file yang dideteksi mengandung tag "Class" kondisi if bernilai true sedangkan kondisi *nested if* bernilai *true.*	Nilai kembalian yang dihasilkan adalah nilai seluruh element XML sesuai kondisi if pada sebuah variabel penampung bertipe *array list* dengan tipe data *String.*	Nilai kembalian yang dihasilkan adalah nilai seluruh element XML sesuai kondisi if pada sebuah variabel penampung bertipe *array list* dengan tipe data *String.*	*Valid*

6.1.3 Pengujian Unit *Method* OpenFileFrom

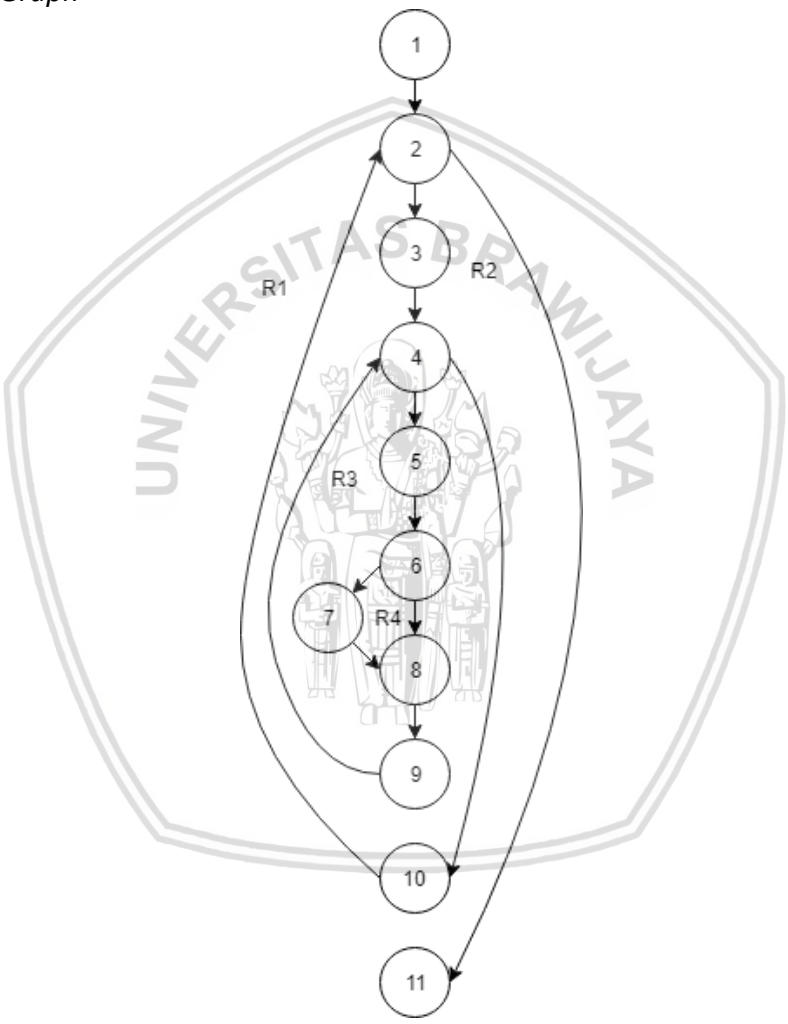
1. *Pseudocode*

No	Pseudocode
1	`Buat array penampung hasil keluaran sistem`
2	`for(setiap value pada method readGeneralizationFrom) then`
3	`inisialisasi obyek fXmlFile dengan path dokumen output dari masukan sistem setelah disubmit agar dapat membuka setiap class yang akan diidentifikasi`
3	`Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya`
3	`File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag`

	"Operation" pada xml file nya
4	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
5	inisialisasi setiap isi dokumen kedalam sebuah Node
6	if(Node := semua element dari xml)then
7	inisialisasi setiap isi Node kedalam obyek element
7	inisialisasi nilai tersebut kedalam array penampung hasil pencarian temp dengan nilai elementnya yang memiliki atribut "Name".
8	end if
9	end for
10	end for
11	kembalikan nilai filefrom sebagai return pada operasi openFileFrom

2. *Basis Path Testing*

2.1 Flow Graph



Gambar 6.2 *Flow Graph Method* OpenFileFrom

2.2 Cyclomatic Complexity (V(G))

- V(G) = Jumlah Region = 4
- V(G) = Jumlah edge – jumlah node + 2 = (13 – 11) + 2 = 4
- V(G) = Jumlah predicate node + 1 = 3 + 1 = 4

2.3 Independent Path

- Jalur 1 : 1-2-11
- Jalur 2 : 1-2-3-4-5-6-8-9-4-10-2-11
- Jalur 3 : 1-2-3-4-5-6-7-8-9-4-10-2-11

- Jalur 4 : 1-2-3-4-10-2-11

Tabel 6.2 Hasil pengujian unit *method* OpenFileFrom

No	No. Jalur	Data Pengujian	Expected Result	Result	Status
1	1	Jika kondisi pada *looping* bernilai *false.*	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*
2	2	Jika file yang dideteksi mengandung tag “Operation” namun kondisi if bernilai *false.*	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*
3	3	Jika file yang dideteksi mengandung tag “Operation” dan kondisi if bernilai true.	Nilai kembalian yang dihasilkan adalah nilai seluruh element XML sesuai kondisi if pada sebuah variabel penampung bertipe *array list* dengan tipe data *String.*	Nilai kembalian yang dihasilkan adalah nilai seluruh element XML sesuai kondisi if pada sebuah variabel penampung bertipe *array list* dengan tipe data *String.*	*Valid*
4	4	Jika file yang dideteksi tidak mengandung tag “Operation”.	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*

6.1.4 Pengujian Unit *Method* OpenFileTo

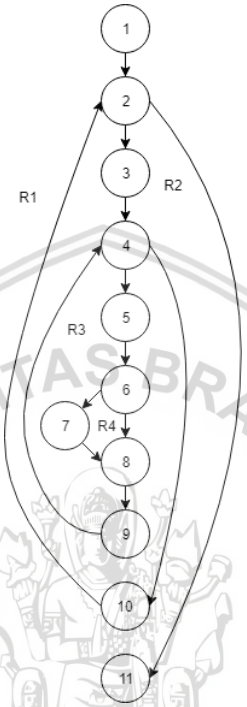
1. *Pseudocode*

No	Pseudocode
1	Buat array penampung hasil keluaran sistem
2	for(setiap value pada method readGeneralizationTo) then
3	inisialisasi obyek fXmlFile dengan path dokumen output dari masukan sistem setelah disubmit agar dapat membuka setiap class yang akan diidentifikasi
3	Obyek fXmlFile dilakukan *parsing* kedalam bentuk dokumen agar dapat diidentifikasi setiap tag filenya
3	File yang telah menjadi dokumen dilakukan pencarian berdasarkan tag “Operation” pada xml file nya
4	for (int i = 0 dimana i hingga nList.getLength() dan i bertambah 1 setiap loop) then
5	inisialisasi setiap isi dokumen kedalam sebuah Node
6	if(Node := semua element dari xml)then
7	inisialisasi setiap isi Node kedalam obyek element

```
7   inisialisasi nilai tersebut kedalam array penampung hasil pencarian
    temp dengan nilai elementnya yang memiliki atribut "Name".
8   end if
9   end for
10  end for
11  kembalikan nilai filefrom sebagai return pada operasi openFileTo
```

2. *Basis Path Testing*

2.1 Flow Graph



Gambar 6.3 *Flow Graph Method* OpenFileTo

2.2 Cyclomatic Complexity (V(G))

- V(G) = Jumlah Region = 4
- V(G) = Jumlah edge – jumlah node + 2 = (13 – 11) + 2 = 4
- V(G) = Jumlah predicate node + 1 = 3 + 1 = 4

2.3 Independent Path

- Jalur 1 : 1-2-11
- Jalur 2 : 1-2-3-4-5-6-8-9-4-10-2-11
- Jalur 3 : 1-2-3-4-5-6-7-8-9-4-10-2-11
- Jalur 4 : 1-2-3-4-10-2-11

Tabel 6.3 Hasil pengujian unit *method* OpenFileTo

No	No. Jalur	Data Pengujian	Expected Result	Result	Status
1	1	Jika kondisi pada *looping* bernilai *false.*	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*
2	2	Jika file yang dideteksi mengandung	Nilai kembalian yang dihasilkan	Nilai kembalian yang dihasilkan adalah null.	*Valid*

		tag “Operation” namun kondisi if bernilai *false*.	adalah null.		
3	3	Jika file yang dideteksi mengandung tag “Operation” dan kondisi if bernilai true.	Nilai kembalian yang dihasilkan adalah nilai seluruh element XML sesuai kondisi if pada sebuah variabel penampung bertipe *array list* dengan tipe data *String.*	Nilai kembalian yang dihasilkan adalah nilai seluruh element XML sesuai kondisi if pada sebuah variabel penampung bertipe *array list* dengan tipe data *String.*	*Valid*
4	4	Jika file yang dideteksi tidak mengandung tag “Operation”.	Nilai kembalian yang dihasilkan adalah null.	Nilai kembalian yang dihasilkan adalah null.	*Valid*

6.2 Pengujian Integrasi

Pada pengujian ini merupakan pengujian yang dilakukan dengan berfokus kepada desain serta konstruksi dari aristektur perangkat lunak. Tabel 6.3 berikut merupakan identifikasi serta rencana uji yang akan dilakukan dalam pengujian integrasi pembangunan sistem untuk pendeteksian *code smells refused bequest*. Berikut penjelasan mengenai pengujian yang akan dilakukan pada Tabel 6.3.

Tabel 6.4 Identifikasi dan Rancangan Pengujian Integrasi

No	Nama Kelas	Nama *Method*	Tujuan
1	Readfile	openFileFrom(String, int)	Melakukan deteksi *refused bequest* dengan menggunakan masukan berupa nama *file project* dengan tipe data xml dan nilai *index* berdasarkan hubungan *Generalization* yang telah dideteksi.
	DetectL0_4	DetectRefused(String, int)	
2	Readfile	openFileTo(String, int)	Melakukan deteksi *refused bequest* dengan menggunakan masukan berupa nama *file project* dengan tipe data xml dan nilai *index* berdasarkan hubungan *Generalization* yang telah dideteksi.
	DetectL0_4	DetectRefused(String, int)	

3	Controller_XML	write(String)	Melakukan deteksi file *project* dan memisahkan berdarkan *tag class* yang telah dideteksi.
	Readfile	Readfile(String)	

Tabel 6.5 Hasil Pengujian Integrasi Nomor 1

No. Uji	1
Input Pertama	FILENAME = "D:\\class\\project.xml" indexRealize = 3
Method dari kelas Readfile	openFileFrom(String, int)
Output Pertama / Input Kedua	Generalization 1 = setLevel(3.0) Generalization 2 = setLevel(3.0) Generalization 3 = setLevel(3.0)
Method dari kelas DetectL0_4	DetectRefused(String, int)
Expected Result	Hasil deteksi pada setiap hubungan *Generalization* diberikan level untuk ditampilkan pada *thermometer smells*.
Result	Hasil deteksi pada setiap hubungan *Generalization* diberikan level untuk ditampilkan pada *thermometer smells*.
Status	Valid

Tabel 6.6 Hasil Pengujian Integrasi Nomor 2

No. Uji	2
Input Pertama	FILENAME = "D:\\class\\project.xml" indexRealize = 3
Method dari kelas Readfile	openFileTo(String, int)
Output Pertama / Input Kedua	Generalization 1 = setLevel(3.0) Generalization 2 = setLevel(3.0) Generalization 3 = setLevel(3.0)
Method dari kelas DetectL0_4	DetectRefused(String, int)

Expected Result	Hasil deteksi pada setiap hubungan *Generalization* diberikan level untuk ditampilkan pada *thermometer smells*.
Result	Hasil deteksi pada setiap hubungan *Generalization* diberikan level untuk ditampilkan pada *thermometer smells*.
Status	Valid

Tabel 6.7 Hasil Pengujian Integrasi Nomor 3

No. Uji	3
Input Pertama	FILENAME = "D:\\class\\project.xml"
Method dari kelas Readfile	write(String)
Output Pertama / Input Kedua	fl = File(dengan path yang ditentukan dan nama file berdasarkan id *class* yang dideteksi).
Method dari kelas DetectL0_4	Readfile(String)
Expected Result	File xml yang dimasukkan dipisahkan berdasarkan *tag class* dan disimpan pada *folder project*.
Result	File xml yang dimasukkan dipisahkan berdasarkan *tag class* dan disimpan pada *folder project*.
Status	Valid

6.3 Pengujian Validasi

Pengujian validasi merupakan suatu proses untuk menguji seluruh kebutuhan yang telah ditentukan pada analisis kebutuhan telah dibangun dan telah sesuai dengan skenario yang dibuat. Pengujian validasi disebut berhasil ketika seluruh sistem yang telah dibuat berdasarkan kebutuhan yang telah ditentukan sesuai dengan skenario yang telah ditetapkan.

Tabel 6.8 Pengujian Validasi Memilih File XML

Kode Kebutuhan	CSRB-F-001
Nama Kasus Uji	Memilih File XML
Prosedur	1. Menekan tombol *choose file* 2. Sistem menampilkan *open file dialog* untuk *user* memilih file masukan berupa file xml

	3. *User* memilih sebuah file xml, kemudian memilih tombol open
Expected Result	Sistem menampilkan notifikasi berupa nama file dan ekstensi pada *text field*.
Result	Sistem menampilkan notifikasi berupa nama file dan ekstensi pada *text field*.
Status	Valid

Tabel 6.9 Pengujian Validasi Memasukkan File XML

Kode Kebutuhan	CSRB-F-002
Nama Kasus Uji	Memasukkan File XML
Prosedur	1. Menekan tombol *submit* ketika file telah dipilih dalam sistem 2. Sistem melakukan submit file dengan ekstensi XML kedalam sistem 3. Sistem menampilkan notifikasi *file submitted*
Expected Result	Sistem berhasil melakukan submit file XML dan menampilkan notifikasi.
Result	Sistem berhasil melakukan submit file XML dan menampilkan notifikasi.
Status	Valid

Tabel 6.10 Pengujian Validasi Menghapus File XML

Kode Kebutuhan	CSRB-F-003
Nama Kasus Uji	Menghapus File XML
Prosedur	1. Menekan tombol *remove* ketika file telah dipilih dalam sistem 2. Sistem menghapus file dengan ekstensi XML kedalam sistem 3. Sistem menampilkan notifikasi *file has been removed*
Expected Result	Sistem berhasil menghapus file pilihan *user* dan menampilkan notifikasi.
Result	Sistem berhasil menghapus file pilihan *user* dan

	menampilkan notifikasi.
Status	Valid

Tabel 6.11 Pengujian Validasi Mendeteksi File XML

Kode Kebutuhan	CSRB-F-004
Nama Kasus Uji	Mendeteksi File XML
Prosedur	1. Menekan tombol *detect file* setelah melakukan *submit* 2. Sistem melakukan deteksi dan menampilkan notifikasi *file has been checked*
Expected Result	Sistem berhasil melakukan deteksi dan menampilkan notifikasi.
Result	Sistem berhasil melakukan deteksi dan menampilkan notifikasi.
Status	Valid

Tabel 6.12 Pengujian Validasi Melihat Ringkasan Hasil Deteksi File XML

Kode Kebutuhan	CSRB-F-005
Nama Kasus Uji	Melihat Ringkasan Hasil Deteksi File XML
Prosedur	1. Menekan tombol *view summary* setelah melakukan deteksi file xml. 2. Sistem menampilkan ringkasan hasil deteksi berupa nama *project,* jumlah *class,* jumlah *Generalization* dan jumlah *interface*.
Expected Result	Sistem berhasil menampilkan ringkasan hasil deteksi file XML.
Result	Sistem berhasil menampilkan ringkasan hasil deteksi file XML.
Status	Valid

Tabel 6.13 Pengujian Validasi Melihat Kategori File XML

Kode Kebutuhan	CSRB-F-006
Nama Kasus Uji	Melihat Kategori File XML

Prosedur	1. Menekan tombol *view category* setelah melakukan deteksi file xml. 2. Sistem menampilkan kategori file berdasarkan hubungan *Generalization* yang telah dilakukan deteksi *refused bequest*. 3. Memilih salah satu kategori file xml. 4. Sistem menampilkan *thermometer smells* berdasarkan hasil deteksi sistem.
Expected Result	Sistem berhasil menampilkan *thermometer smells* berdasarkan kategori yang dipilih.
Result	Sistem berhasil menampilkan *thermometer smells* berdasarkan kategori yang dipilih.
Status	Valid

Tabel 6.14 Pengujian Validasi sistem pendeteksian dengan batas waktu

Kode Kebutuhan	CSRB-NF-001
Nama Kasus Uji	Sistem dapat melakukan proses deteksi refused bequest dengan batas waktu respon sistem dibawah 10 detik.
Prosedur	1. Melakukan pendeteksian *file project* secara normal pada sistem.
Expected Result	Sistem berhasil melakukan deteksi *file xml* kurang dari 10 detik.
Result	Sistem berhasil melakukan deteksi *file xml* kurang dari 10 detik.
Status	Valid

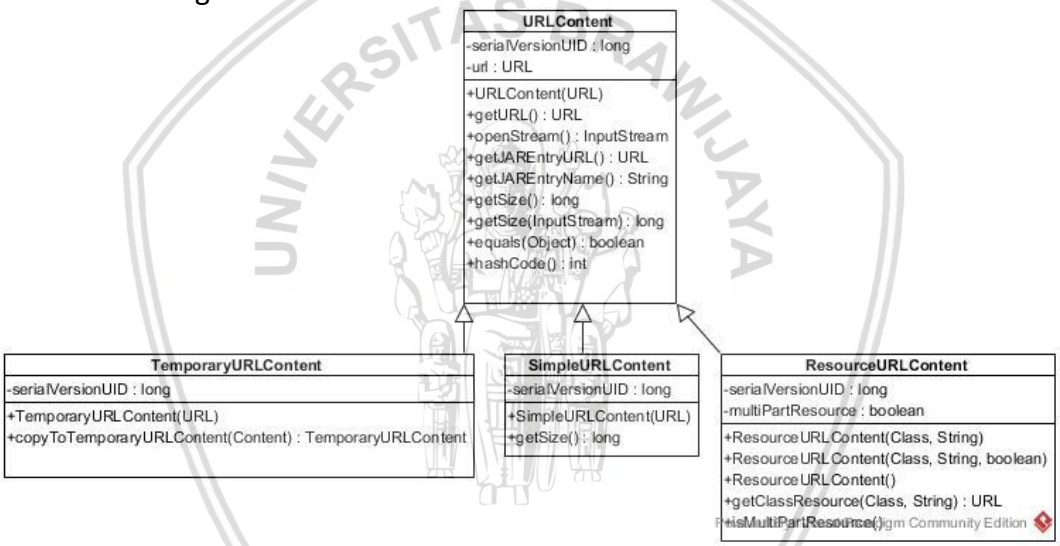
6.4 Pembahasan Hasil Pengujian Sistem

Pengujian yang telah dilakukan untuk menguji sistem pendeteksian *code smells refused bequest* antara lain adalah pengujian unit, pengujian integrasi, pengujian validasi. Berdasarkan pengujian yang telah dilakukan, didapatkan hasil status valid untuk seluruh kasus uji, sehingga sistem yang dibangun dapat dijadikan untuk sistem kakas bantu pendeteksian *code smells refused bequest*. Pengujian yang dilakukan menggunakan metode *whitebox testing basis path testing* dengan menguji 3 method utama dari sistem yaitu *method* "Readfile(String)" dari kelas Readfile, *method* "openFileFrom(String, int)" dari kelas Readfile, dan *method* "openFileFrom(String, int)" dari kelas Readfile.

Pengujian unit yang dilakukan untuk semua method pada pengujian unit menghasilkan beberapa kasus uji yang mana semuanya berstatus valid. Pengujian integrasi yang dilakukan untuk semua method pengujian integrasi menghasilkan status valid. Pada pengujian validasi yang dilakukan, yaitu melakukan pengujian pada seluruh scenario untuk kebutuhan fungsional termasuk alternatif kondisi yang ada. Kebutuhan fungsional beserta alternatifnya menghasilkan hasil yang baik yaitu keseluruhan kasus uji bernilai status valid adapun untuk kebutuhan non-fungsional menghasilkan 2 kasus uji yang keseluruhannya bernilai status valid.

6.5 Pengujian Aplikasi SweetHome3D

Pada bagian ini akan dijelaskan mengenai sistem yang akan diuji pada penelitian ini. Pada metodologi penelitian telah disampaikan bahwa sistem yang akan diuji adalah aplikasi *Java* SweetHome3d yang merupakan aplikasi interior desain rumah bersifat *open source*. *Class diagram* yang akan diuji pada penelitian ini adalah sebagai berikut:



Gambar 6.4 Class Diagram SweetHome3D

Pada Gambar 6.4 merupakan *class diagram* yang diuji pada sistem pendeteksian *code smells refused bequest* dimana terdapat 3 hubungan generalisasi. Pada pengujian yang dilakukan, sistem pendeteksi melakukan proses deteksi *code smells* dengan melihat kepada hubungan generalisasi yang ada. Sehingga pada hasil pendeteksian didapatkan untuk pengujian hubungan generalisasi pertama yaitu antara *class URLContent* dan *TemporaryURLContent* adalah level 3 pada termometer smells yang dapat diartikan bahwa hal ini menunjukkan bahwa kemunculan *refused bequest* bersifat kuat. Pada hubungan generalisasi kedua, yaitu antara *class URLContent* dan *SimpleURLContent* adalah level 2 pada termometer smells yang dapat diartikan bahwa hal ini menunjukkan bahwa kemunculan *refused bequest* bersifat relatif lebih lemah dari hubungan generalisasi pertama. Pada hubungan generalisasi ketiga, yaitu antara *class URLContent* dan *ResourceURLContent* adalah level 3 pada termometer smells yang dapat diartikan bahwa kemunculan *refused bequest* bersifat kuat.

Penjelasan mengenai tingkatan hasil deteksi dapat dilihat pada tinjauan pustaka dilaporan penelitian ini.

Berdasarkan pengujian pada aplikasi yang telah dilakukan dipenelitian ini, dapat dibuat sebuah kesimpulan bahwa aplikasi SweetHome3D terdeteksi memiliki *refused bequest* pada beberapa hubungan generalisasi yang ada pada aplikasi tersebut. Hasil pengujian yang dilakukan pada aplikasi SweetHome3D juga memiliki kemiripan hasil deteksi dengan hasil penelitian yang dilakukan oleh Ligu (2013) dalam jurnalnya. Sehingga sistem pendeteksian *code smells refused bequest* dapat dijadikan sebagai sistem yang dapat digunakan untuk mendeteksi *refused bequest* pada tahap perancangan agar pendeteksian dapat dilakukan sedini mungkin.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan mengikuti fase pengembangan perangkat lunak yaitu analisis kebutuhan sistem, perancangan, implementasi serta pengujian yang dilakukan, maka didapatkan kesimpulan sebagai berikut:

1. Berdasar kepada hasil analisis kebutuhan yang dilakukan, pembangunan sistem untuk pendeteksian *code smells refused bequest* memiliki 6 kebutuhan fungsional dan 1 kebutuhan non-fungsional. Kebutuhan tersebut yang digunakan untuk melakukan proses pendeteksian *code smells* dengan jenis *refused bequest*. Kebutuhan fungsional tersebut didapatkan dari jurnal utama oleh Elvis Ligu dengan judul *Identification of Refused Bequest Code Smells* sebagai dasar penelitian yang dilakukan dan dikembangkan. Pada penelitian ini telah dikembangkan proses mendeteksi *code smells* dengan jenis *refused bequest* pada fase perancangan sistem yaitu saat melakukan perancangan class diagram. Dengan pengembangan ini diharapkan dapat lebih cepat mengetahui *refused bequest* saat melakukan perancangan sehingga sistem yang dikembangkan kemudian dapat terhindar dari *code smells* yang dapat merugikan sistem tersebut. Setelah kebutuhan didapatkan, maka kebutuhan dimodelkan kedalam pemodelan-pemodelan kebutuhan yang dapat berguna untuk pemahaman terhadap sistem yang akan dibangun, adapun pemodelan tersebut yaitu pemodelan *use case diagram,* dan *user case scenario*.
2. Berdasarkan perancangan sistem yang telah dilakukan, didapatkan perancangan arsitektur, perancangan komponen, dan perancangan antarmuka untuk membangun sistem pendeteksian *code smells* dengan jenis *refused bequest*. Perancangan tersebut didapatkan dari hasil analisis kebutuhan yang dilakukan sebelumnya. Adapun pada perancangan arsitektur didapatkan rancangan *sequence diagram* dan rancangan *class diagram*. Pada perancangan komponen didapatkan rancangan algoritma-algoritma utama dari sistem yang akan dibangun. Sedangkan pada bagian perancangan antarmuka berisi rancangan *layout* sistem mulai dari halaman utama sistem, menu *analyze*, menu *result,* hingga menu *about.*
3. Berdasarkan implementasi sistem yang telah dilakukan, didapatkan hasil implementasi yang mengacu kepada fase perancangan sebelumnya antara lain adalah spesifikasi sistem, implementasi kode program, dan implementasi antar muka sistem. Pada spesifikasi sistem terdapat penjelasan tentang perangkat keras yang digunakan saat membangun sistem, perangkat lunak yang digunakan, serta sistem operasi yang digunakan untuk melakukan pembangunan dan pengujian sistem pendeteksian *code smells* dengan jenis *refused bequest*. Adapun pada bagian implementasi kode program mengacu kepada perancangan komponen sebagai dasar alur logika sistem dalam mendeteksi *refused bequest*. Serta pada implementasi antarmuka sistem diperoleh tampilan antarmuka sistem dengan menggunakan bahasa pemrograman *Java Swing*. Implementasi antarmuka yang dilakukan mengacu

kepada perancangan antarmuka yang dilakukan sebelumnya dengan tujuan agar memudahkan interaksi antara sistem dan pengguna sistem.

4. Pengujian yang telah dilakukan untuk menguji sistem pendeteksian *code smells refused bequest* antara lain adalah pengujian unit, pengujian integrasi, pengujian validasi. Berdasarkan pengujian yang telah dilakukan, didapatkan hasil status valid untuk seluruh kasus uji, sehingga sistem yang dibangun dapat dijadikan untuk sistem kakas bantu pendeteksian *code smells refused bequest*. Pengujian yang dilakukan menggunakan metode *whitebox testing basis path testing* dengan menguji 3 method utama dari sistem yaitu *method* "Readfile(String)" dari kelas Readfile, *method* "openFileFrom(String, int)" dari kelas Readfile, dan *method* "openFileFrom(String, int)" dari kelas Readfile. Pengujian unit yang dilakukan untuk semua method pada pengujian unit menghasilkan beberapa kasus uji yang mana semuanya berstatus valid. Pengujian integrasi yang dilakukan untuk semua method pengujian integrasi menghasilkan status valid. Pada pengujian validasi yang dilakukan, yaitu melakukan pengujian pada seluruh scenario untuk kebutuhan fungsional termasuk alternatif kondisi yang ada. Kebutuhan fungsional beserta alternatifnya menghasilkan hasil yang baik yaitu keseluruhan kasus uji bernilai status valid adapun untuk kebutuhan non-fungsional menghasilkan 2 kasus uji yang keseluruhannya bernilai status valid.

7.2 Saran

Berdasarkan hasil penelitian yang dilakukan, didapatkan beberapa saran yang diberikan untuk pengembangan sistem pendeteksian *code smells refused bequest* ini selanjutnya, antara lain adalah:

1. Sistem dapat lebih dikembangkan dalam hal pendeteksian *code smells* dengan jenis *refused bequest* berdasarkan fase pengembangan perangkat lunak lainnya.
2. Sistem dapat dibuat lebih menarik agar sistem menjadi lebih baik lagi kedepannya.
3. Untuk kedepannya sistem dapat dikembangkan kedalam website agar lebih memudahkan akses sistem dan kegunaan sistem lebih luas.

DAFTAR PUSTAKA

Aristyagama, Y. H, 2016. *Framework Deteksi Bad Smell Code Semi Otomatis untuk Pemrograman Tim*. Bandung: Magister Teknik Elektro TMDG'09.

Fontana, F. A., 2011. An experience report on using code smells detection tools. *Fourth International Conference on Software Testing,* pp. 450-457.

Gaur, Jai. 2016. *A Walk Through of Software Testing Techniques.* India:Amity University.

Kneuper, R. 2017. *Sixty Years of Software Development Life Cycle Models. IEEE International, pp. 41-54.*

Ligu, E., 2013. Identification of Refused Bequest Code smells. *IEEE International,* pp. 392-395.

M. Fowler, K. Beck, J. Brant, W. Opdyke dan d. Roberts, *Refactoring: Improving the Design of Existing Code*, Boston: Addison-Wesley Longman Publishing Co., Inc, 2000.

Pressman, R., 2001. *Software Engineering: A Practitioner's Approach 5.* penyunt. New York: McGraw-Hill.

Pressman, R., 2010. *Software Engineering: A Practitioner's Approach* 7. penyunt. New York: McGraw-Hill.

Putro, H. P dkk, 2010. *Deteksi Code Smell Pada Kode Program Dalam Representasi AST Dengan Pendekatan By Rules. Bandung: Kelompok Keahlian RPL dan Data Institut Teknologi Bandung.*

Rumbaugh, James et al., 2005. *The Unified Modeling Language Reference Manual.* 2*nd ed.* Boston:Addison-Wesley.

Runeson, Per. 2011. *Software Testing.* Software Engineering Research Group. Lund University.

S. M. Olbrich, D. S. Cruzes dan D. I. Sjberg, "Are all Code Smells Harmful? A Study of God *Class*es and Brain *Class*es inthe Evolution of three Open Source Systems," *Software Maintenance (ICSM), 2010 IEEE International Conference,* pp. 1 - 10, 2010.

Silva, L. H., 2015. *VI Brazilian Conference on Software: Theory and Practice (Tools Track).* Chile, Department of Computer Science.

Sommervile, Ian. 2011. *Software Engineering 9th Edition.* US America: Pearson Education, Inc.

Waldo, Jim. 2006. *On System Design*. *Sun Labs,* pp. 1-16.

X. Zhao, X. Xuan dan S. Li, "An Empirical Study of Long Method and God Method in Industrial Projects," *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop,* pp. 109 - 114, 2015.